mesytec **MDPP-32** is a fast high resolution time an amplitude digitizer. It is internally realised as a 32 channel adjustable low noise amplifier and a variable differentiation stage, followed by filters and 80 MHz sampling ADCs. The digitized data are analyzed in an FPGA and reconstructed with highest precision. This allows to achieve unique timing and amplitude resolution.

Hardware features:

- **Low noise variable gain input amplifiers.**
  - Input signals for maximum range (highest spectrum channel) from **1.5 mV to 20 V**.
  - Input noise down to **2 µV** @ 2 µs shaping.
- **Variable hardware pre-differentiation**
  Allows large offsets and signal stacking without effect on the amplitude or timing resolution.
- **Gain-polarity jumpers**
  determine: termination, polarity, input range and input configuration (differential / unipolar).
- **Two high resolution monitor outputs**
  for monitoring internal signals and noise via oscilloscope.
- **Up to 3 software modules**
  can be stored on board and can be selected by switch or VME.
- **Installation and update via USB or VME**
- **VME64 and VME64-X compatible.**
  can use VME64-X geo addressing

Software modules

Large digital resources allow precise wave form reconstruction.

- timing down to **75 ps rms**
- amplitude resolution better than **16k**.
- Trigger threshold down to **1/2000** of maximum range.

Software modules:

- Amplitude & time for standard preamps (SCP), 32 k/75 ps
- QDC: pulse shape discrimination, Two integration times. Charge & time , self gating, 4 k/75 ps

Planned module:

- Peak sensing ADC, 16 k, self gating or external

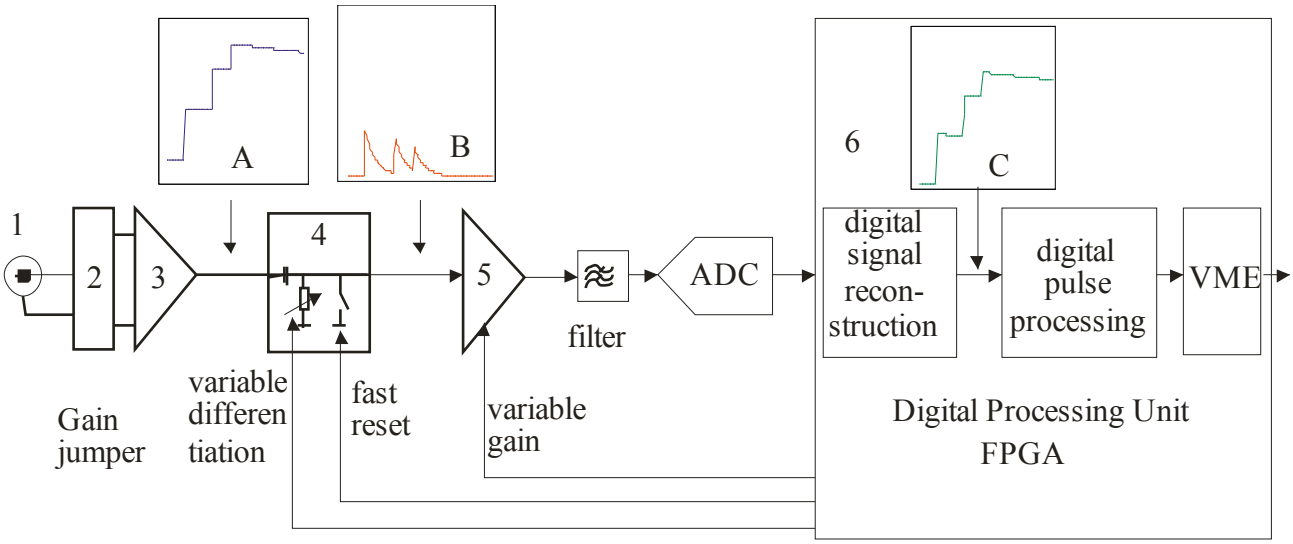
mesytec GmbH & Co. KG
Wernher-von-Braun- Str. 1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Hardware concept of MDPP-32

MDPP-32 was developed to meet the following challenges:

1. **Easy to use:**
   - No knowledge of the internal signal processing required.
   - Only essential signal parameters and settings required for operation.
   - Input is capable to directly accept any preamplifier signal.
2. **Works together with existing VME modules -** accepts and creates triggers for an external experiment logic.
3. **Provides very good timing -** good enough to replace external CFDs and TDCs for most applications.
4. **Amplitude resolution as good as best analog solutions,** including ballistic loss correction, pile up rejection, baseline restoration.

To meet those goals, a new hardware concept was required as shown in the following figure.



The input (1) was designed to allow two configurations:

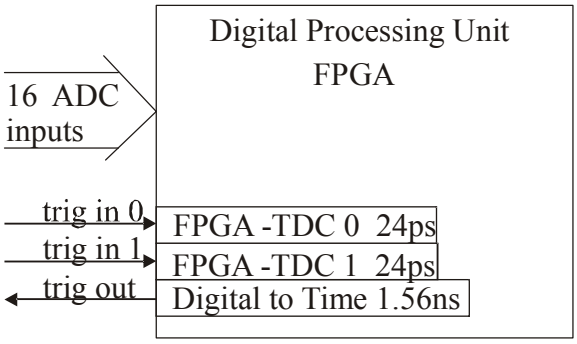
- differential and unipolar input with a standard 34 pin header connector (depending on applied jumpers).
- Unipolar input with Lemo inputs.

The input signal amplitude for maximum output range is from 1.5 mV to 20 V and can be set by different **input jumpers (2),** a **variable gain stage (5)** (gain 1 to 24) and by additional scaling of the **digitized data (6)**. So a continuous gain from 1.00 to 200.00 is provided for each gain jumper set.

The input is followed by a high dynamic range, **low noise amplifier (3).** Its signal is then differentiated by an adjustable **differentiation stage (4)**. It is part of the shaping filter, and is set by the central logic unit. It also includes a fast reset circuit, which allows fast recovery from large overflow and underflow signals. It delivers an output which is **free of offset (B)** and eliminates the typical **stacking (A)** of charge integrating preamplifiers. So the dynamic range of the ADC can be fully used.

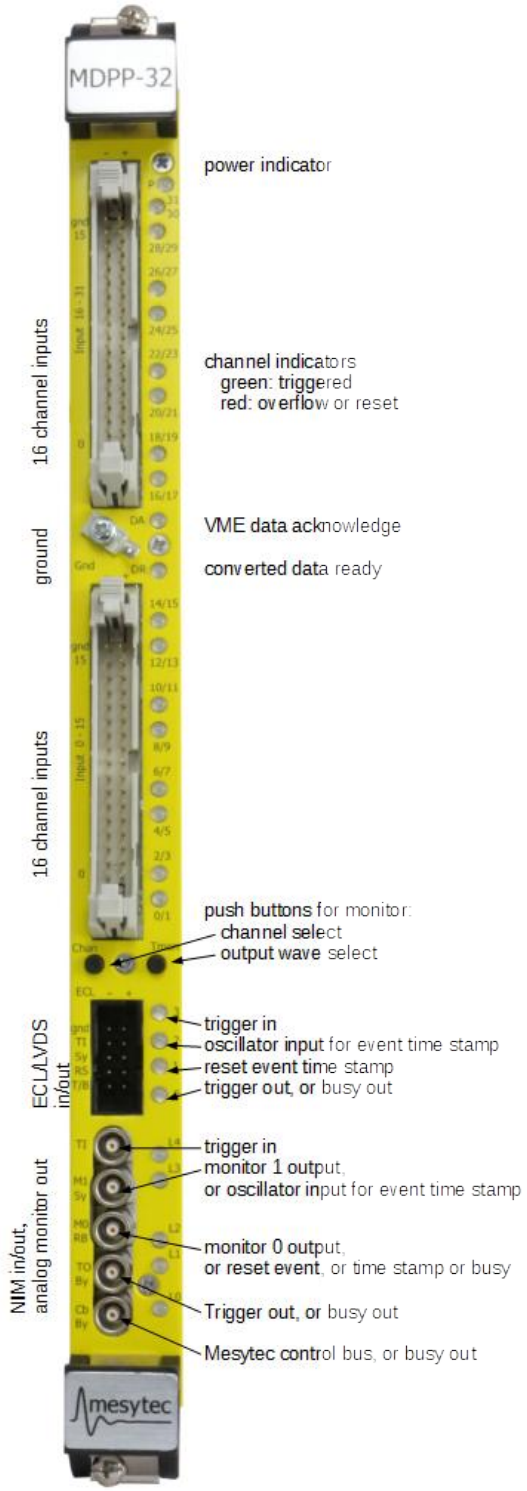
The **digital processing unit (6)** can be loaded with different software programs (up to 4 may be stored on board). With "SCP" software module for processing of charge sensitive preamp signals, the ADC signal is regenerated by an integration and the signal input (without offset) is fully recovered (C). Then the signal is processed with high precision and the help of 180 signal processors. Details of processing are described at another place with the software modules.

The MDPP-32 provides **further resources** to allow easy integration into a standard nuclear physics data acquisition system. It provides two high resolution trigger inputs (24 ps timing resolution). One of them may start the window of interest, while the other may add an additional time mark from an external detector.



Internally created triggers from the 16 input channels can be output to a high resolution trigger output (1.56 ns resolution). This allows to create an external experiment trigger from many modules. The trigger delay is typically 400 ns (+ TF-integration time) from the edge of analog input signal.
The externally created trigger can then be fed to one of the trigger inputs to start the window of interest. There are no problems with delay between trigger and channel input data, because the window of interest can be shifte int time by up to +-25 us.

Front panel elements



MDPP-32 provides 16 bi-color LED indicators for channel pairs. When lighting green the signal is within the legal range, red indicates an under- or overflow. When a channel is selected for monitoring, the LED flashes red.

There are two push buttons on the front panel to control the monitor function. One selects a channel to monitor, the other one selects a pair of wave forms.

The wave forms reflect signals generated in the pulse processing chain inside the FPGA. The signals are output at two Lemo connectors (L2, 3)

The following oscilloscope picture shows a wave form pair:

The green curve shows the differentiated signal. The magenta one shows the integral of the differentiated signal, creating the typical triangular shaping curve.

triangular shaper, differentiated sign

Also strongly amplified wave forms can be selected to monitor the structure of external noise.

Monitor signals: mon0 / mon1

1. raw data / reconstructed signal
2. TF-output / Shaper output
3. Shaper x 32, BLR-signal x 32
4. Shaper x 32, Timing Filter x 32

Timing and data concept - "Window of interest"

The window of interest concept is the same as for MQDC-32 and MTDC-32.

MDPP-32 creates an output trigger from any selected input channel or by a logical "or" of all or some user selectable channel inputs. The output trigger has a timing resolution of maximum 1.56 ns.

It is intended to feed a trigger logic in more complex setups.

MDPP-32 then expects an external gate from external trigger logic, which is evaluated with a high resolution (24 ps) time to digital converter (TDC).

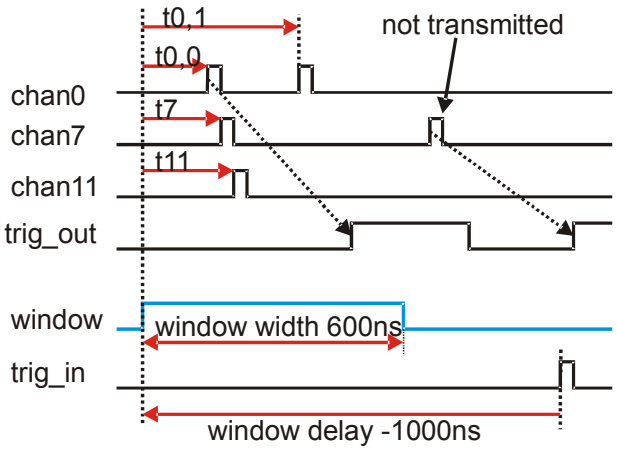
In the simplest configuration, output trigger can be fed back to the trigger input to allow self triggering.

The input trigger can be shifted internally by +/-25 us and starts a **window of interest** with adjustable width (1.5 ns to 25 us). All triggers generated by the CFD discriminators from the channel inputs, which fall into this window, are sent to a large data buffer for read out by The VME bus. The transmitted data are:

- the time difference to the window start (in 24 ps resolution)
- the converted shaper amplitude.
- over/underflow, and pileup flags

The provided buffer structure and all features like Event time stamp, event counter... are identical to all other mesytec VME-modules.

Example:



In the example, the internal triggers of three input channels are shown after the CFD discriminator. Two pulses are detected in channel 0 and 7, one in channel 11. The very first pulse, which is in channel 0, starts the output trigger. This trigger may be fed through an external electronics and - if coincident to other detectors and data acquisition is not busy - returns delayed to the trigger input . It arrives with a fixed delay of 800 ns. So it has to be shifted back in time by MDPP-32 (here 1000 ns) and a coincidence time (window with) of 600 ns has to be created. As can be seen 4 hits fall into this window (2 x channel 0, 1x channel 7 and 11).

So four time differences referenced to the window start are calculated, and are sent to a data buffer together with the according amplitudes.

All software modules will have the following features:

- Support different types of event synchronization stamping (based on VME-clock or external clock)
- Multiplicity filter, selects events in specified multiplicity range
- mesytec control bus to control external mesytec modules
- Address modes: A24 / A32
- Data transfer modes: D16 (registers)
- D32, BLT32, MBLT64, CBLT, CMBLT64
- Multicast for event reset and time stamping start

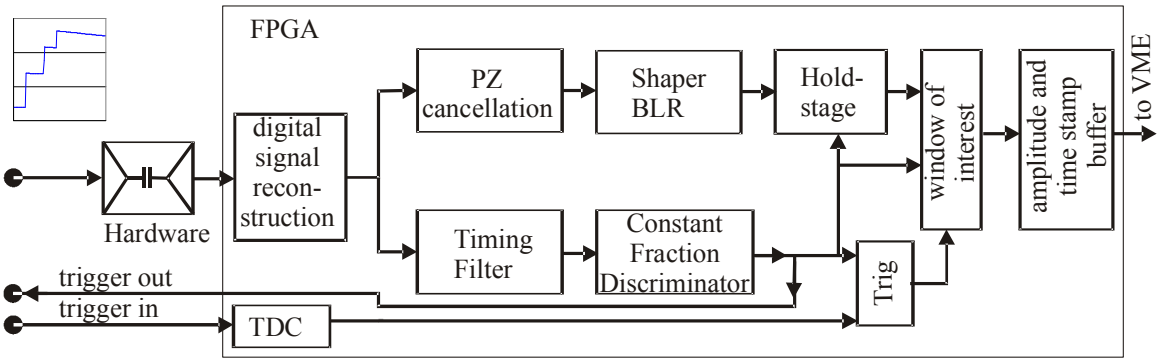
Hardware features

- Live insertion (can be inserted in a running crate)
- Power consumption: 14 W,
  + 5 V, 2 A
  + 12 V, 100 mA
  - 12 V, 200 mA.

Software module: "SCP"

(Delivers timing and amplitude for **standard charge sensitive preamplifier** signals)
Replaces shaper, peak sensing ADC, timing filter amplifier, CFD, and TDC.

The following picture shows a schematic representation of the software:



The signal is amplified filtered digitized and reconstructed as described in the hardware chapter. Then it is split into a timing branch, and enters a **timing filter**. It differentiates and integrated the signal with short adjustable time constant. Then a digital **CFD** (discriminator) calculates an amplitude independent time trigger (=time stamp).
In the other "slow" branch the signal is deconvoluted (**PZ cancellation**) and then enters a filter consisting of a differentiator and integrator forming a filter, which produces a **triangular shaping**. Also a **base line restorer** is implemented.
Then signal then enters a **hold stage** which holds the amplitude at a well defined time, determined by the CFD discriminator.-
Then the amplitude and timing values are filtered by a **window of interest** and stored in a **buffer**.

Short data:

- Amplitude resolution of more than 16 k (14 bit)
- Trigger to channel time resolution of **60 ps rms,** uniform at any delay.
- Channel to channel time resolution of **70 ps rms**, uniform at any delay.
- Trigger input with 24 ps timing resolution
- Dynamic range (trigger 2000:1)
- Independent shaping of timing filter and amplitude branch.
- Shaping width can be set from 50 ns to 25 µs FWHM (= 25 ns to 10 us sigma values) in steps of 12.5 ns.
- Timing filter from 15 ns to 1.6 µs.
- Can be operated self triggered or externally triggered
- Outputs internal raw trigger with 1.5 ns time resolution

As easy to operate as all mesytec modules and fully data compatible.

Only five parameters have to be set:

Signal properties:

1. signal rise time 15 ns to 1.6 µs (= TF integration and diff -time)
2. signal decay time (for PZ) 800 ns to ∞.
3. Gain 1 to 200 in steps of 0.01

User settings:

4. Shaping time: 25 ns to 11 µs (50 ns to 25 µs FWHM)
5. Threshold

Output Data

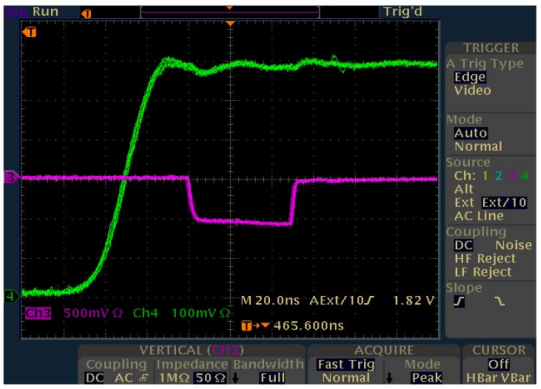
channel 0..15 Amplitude (11 to 16 bit)
channel 16 to 31 time difference to window start (16 bit) 24 ps/chan.

Monitor outputs
(Lemo 2 = mon 0, and Lemo 3 = mon 1)

Switching on the monitor: press pusbutton "chan", then select a wave form with "Tmon" button. The button "chan" allows to switch through the individual channels.

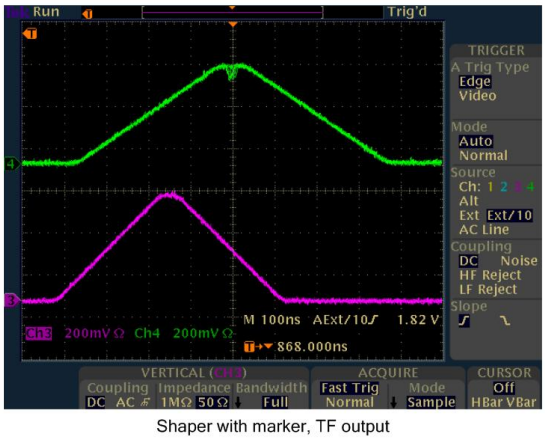
Wave forms:

Tmon 0,
Green: preamplifier signal before ADC,
Magenta: trigger output (Lemo 1)



Input signal, Trigger output

Tmon 1: Check signal **shape and amplitude**
Green: mon 0, triangular shaped signal with flicker mak, showing the sampling time.
Magenta: mon 1, timing filter signal

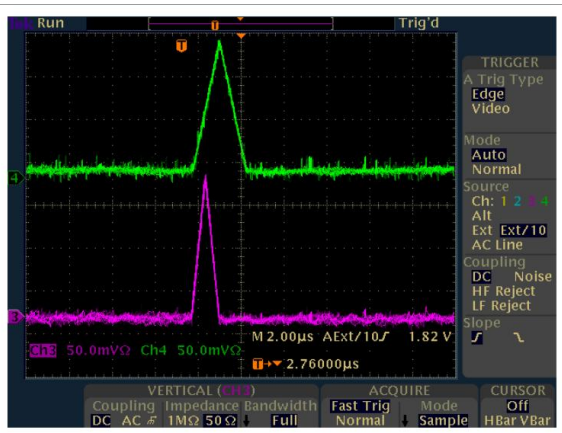


Shaper with marker, TF output

Tmon 2 : Check **pole zero** adjust and **baseline restorer**
Green: mon0, Shaper signal amplified x32.
Magenta: mon 1, reconstructed baseline by the baseline restorer, x 32

Shaper x 32, reconstr. Baseline x 32

Tmon 3: Check **noise**
Green: mon 0, Shaper signal, baseline restored x32
Magenta: mon 1,timing filter signal x 32



Noise inspect, Shaper, Timing Filter

Control input / output

- Differential control inputs:
  - interface any differential signals: ECL, LVDS or LVPECL. They can be individually
  - terminated (110 Ω) via register setting

- NIM inputs:
  - standard NIM, 50 Ω

- NIM output:
  - –0.7 V when terminated with 50 Ω

- mesytec control bus output, shares connector with busy output. +0.7 V terminated

Minimum trigger width for individual inputs is = 10 ns
Maximum external reference synchronisation clock frequency (sync input): 75 MHz

Digital Inputs /outputs (see IO register block 0x6060) and description

MDPP-32 register set, SCP Firmware (processing of Standard Charge sensitive Preamp signals)

Data FIFO, read data at address 0x0000 (access R/W D32, 64)

only even numbers of 32 bit-words will be transmitted. In case of odd number of data words, the last word before EOE word will be a fill word (= 0).
FIFO size: 48 k - 512 = 48640 words with 32 bit length

Header (4 byte)

2 header signature	2 subheader	4	8 module id	3 TDC_resolution → 0x6042	3 ADC_resolution → 0x6046	10 number of following data words, including EOE
b01	b00	xxxx	module id	bxxx	bxxx	number of 32 bit data words

Data (4 byte) DATA event

2 data-sig	2	3	2	1	6	16
b00	01	xxx	(pu, ov)*	Trigger Flag	channel number	ADC value

channel numbers may come in arbitrary order. * pu = pile up flag, ov = overflow or underflow flag

Data (4 byte) DATA event

2 data-sig	2	5	1	6	16
b00	01	xxxxx	Trigger Flag	channel number + 16	TDC time difference

Data (4 byte) Extended time stamp

2 data-sig	2	12	16
b00	10	xxxx xxxx xxxx	16 high bits of time stamp

Data (4 byte), fill dummy (to fill MBLT64 word at odd data number)

2 data-sig	30
b00	0

End of Event mark (4 byte)

2	30
b11	event counter / time stamp

Taking the trigger flag and channel number together, this 6 bit address runs from 0 to 15 for amplitudes, 16 to 31 for time, and 32 / 33 are trigger0 / trigger1 time.
So the full channel address has 6 bits, and runs from 0 to 33. The addresses 32 is for trigger input 0, the address 33 for trigger input 1.

Unlike MDPP-16, MDPP-32 does not support reset preamplifiers

Registers, Starting at address x6000 (access D16)

Address	Name	Bits	dir	default	Comment
	Address registers				
0x6000	address_source	1	RW	0	0 = from board coder, 1 from address_reg
0x6002	address_reg	16	RW	0	address to override decoder on board
0x6004	module_id	8	RW	0xFF	is part of data header If value = FF, the 8 high bits of base address are used (always board coder).
0x6008	soft_reset	1	RW		Write breaks all activities, sets critical parameters to default. **Wait 200ms after writing this register.** Read: MDPP-32 hardware ID: 0x5007
0x600E	firmware_revision	16	R		Example: SCP=2, rev 1.2 = 0x2102 QDC=3, rev 2.1 = 0x3201

	IRQ (ROACK)				
0x6010	irq_level	3	RW	0	IRQ priority 1..7, 0 = IRQ off
0x6012	irq_vector	8	RW	0	IRQ return value
0x6014	irq_test	0	W		initiates an IRQ (for test)
0x6016	irq_reset	0	W		resets IRQ (for test)
0x6018	irq_data_threshold	15	RW	1	Every time the number of 32 bit words in the FIFO exceeds this threshold, an IRQ is emitted. Maximum allowed threshold is "FIFO size".
0x601A	Max_transfer_data	15	RW	1	1) Specifies the amount of **data** read from FIFO before Berr is emitted. Only active for multi **event mode 3**. Transfer is stopped only after full events. Example: At Max_transfer_data = 1, 1 event per transfer is emitted. 2) Specifies the number of **events** read from FIFO before Berr is emitted. Active for multi **event mode 0xb.** Setting the value to 0 allows unlimited transfer.
0x601C	IRQ_source	1	RW	1	IRQ source: 0 = **event** threshold exceeded 1 = **data** threshold exceeded
0x601E	irq_event_threshold	15	RW	1	Every time the number of events in the FIFO exceeds this threshold, an IRQ is emitted.

For multi event mode 2 and 3 the IRQ is:
- **set** when the FIFO fill level gets more than the threshold and is
- **withdrawn** when IRQ is acknowledged or when the fill level goes below the threshold.

	MCST CBLT				
0x6020	cblt_mcst_control	8	RW	0	see table
0x6022	cblt_address	8	RW	0xAA	A31..A25 CBLT- address
0x6024	mcst_address	8	R	0xBB	A31..A25 MCST- address

Bit	Name	Write		Read	
7	MCSTENB	1 0	Enable MCST No effect	0	
6	MCSTDIS	1 0	Disable MCST No effect	1 0	MCST enabled MCST disabled
5	FIRSTENB	1 in a 0	Enable first module CBLT chain No effect	0	
4	FIRSTDIS	1 in a 0	Disable first module CBLT chain No effect	1 0	First module in a CBLT chain Not first module in a CBLT chain
3	LASTENB	1 in an 0	Enable last module CBLT chain No effect	0	
2	LASTDIS	1 in an 0	Disable last module CBLT chain No effect	1 0	Last module in a CBLT chain Not last module in a CBLT chain
1	CBLTENB	1 0	Enable CBLT No effect	0	
0	CBLTDIS	1 0	Disable CBLT No effect	1 0	CBLT enabled CBLT disabled

CBLT Address Field

A31.....A24	A23.....A00
CBLT ADRS	8 high bits, not significant + 16bit module address space

MCST Address Field

A31.....A24	A23.....A00
MCST ADRS	8 high bits, not significant + 16bit module address space

At BLT32

When an empty module is accessed at address 0, BERR is emitted.

At CBLT

When no module contains data, no data are transmitted. The last module emits BERR.

<table>
<tr><td></td><td>FIFO handling</td><td></td><td></td><td></td><td></td></tr>
<tr><td>0x6030</td><td>buffer_data_length</td><td>16</td><td>R</td><td></td><td>amount of data in FIFO
(only fully converted events).
Units → data_len_format.
Can be used for single- and multi event transfer</td></tr>
<tr><td>0x6032</td><td>data_len_format</td><td>2</td><td>RW</td><td>2</td><td>0 = 8 bit, 1 = 16 bit, 2 = 32 bit, 3 = 64 bit,
4= show number of events in FIFO.
The number of 32 bit words is always even.
If necessary the fill word „0“ is added.
For len 0 and 1 the max value 0xFFFF is shown when number exceeds the 16 bit format. The FIFO is not affected.</td></tr>
<tr><td>0x6034</td><td>readout_reset</td><td></td><td>W</td><td></td><td>At single event mode (multi event = 0): allow new trigger, allow IRQ
At multi event = 1: checks threshold, sets IRQ when enough data. Allows safe operation when buffer fill level does not go below the data threshold at readout.
At multievent = 3 : clears Berr, allows next readout</td></tr>
<tr><td>0x6036</td><td>multi event</td><td>4</td><td>RW</td><td>0</td><td>
<table>
<tr><th>Bit[3]</th><th>Bit[2]</th><th>Bit[1:0]</th></tr>
<tr><td>count events
not words
(reg. 0x601A)</td><td>skip berr,
send EOB</td><td>mode[1:0]</td></tr>
</table>
Allow multi event buffering (bit 0, 1)
mode = 0 → no (0x6034 clears event, allows new conversion)
mode = 1 → yes, unlimited transfer, no readout reset required (0x6034 can be written after block readout). Don´t use for CBLT
mode = 3 → yes but MDPP transfers limited amount of data. With reg 0x601A the number of data words can be specified. After word limits is reached, the next end of event mark terminates transfer by emitting Berr. So 0x601A = 1 means event by event transfer (Berr after each event). The next data block can be transferred after writing 0x6034 (resets Berr).

Berr handling: when bit[2] is set:
Send EOB = bit[31:30] = bx10 instead of Berr

Bit[3]: Compare number of transmitted events (not words!) with max_transfer_data (0x601A) for Berr condition.
</td></tr>
<tr><td>0x6038</td><td>marking_type</td><td>2</td><td>RW</td><td>0</td><td>00 → event counter
01 → time stamp
11 → extended time stamp

→ next page</td></tr>
</table>

0x603A	start_acq	1	RW	1	1 → start accepting triggers If no external trigger logic, which stops the gates when daq is not running, is implemented, this register should be set to 0 before applying the FIFO_reset to get a well defined status. When setting it to 1 again for data acquisition start, the module is in a well defined status.
0x603C	FIFO_reset		W		Initialize FIFO
0x603E	data_ready	1	R		1 → data available

	operation mode				
0x6042	tdc_resolution	3	RW	5	5 → 781 ps = 25 ns / 32 4 → 391 ps = 25 ns / 64 3 → 195 ps = 25 ns / 128 2 → 98 ps = 25 ns / 256 1 → 49 ps = 25 ns / 512 0 → 24 ps = 25 ns / 1024
0x6044	output_format	2	RW	0	0 = standard (time and amplitude) 1 = amplitude only 2 = time only
0x6046	adc_resolution	3	RW	4	number of valid output bits for amplitude 0 = 16 bits 1 = 15 bits 2 = 14 bits 3 = 13 bits 4 = 12 bits

	Trigger				
0x6050	win_start	15	RW	16k-16	Unit: 25ns/16 = 1.56 ns Start window of interest: 0x0000 start at -25.56us 0x7FFF start at +25.56us 0x4000 = 16k no delay < 16 k, window starts before Trigger > 16 k, window is delayed
0x6054	win_width	14	RW	32	Unit: 1.56 ns, max 16 k = 25.56 us
0x6058	trig_source	10		0x100	**Defines the trigger which creates the window of interest**. This can be: one or both of the trigger inputs, lower 16 channels(B0), upper 16 channels(B1), or both banks (all channels).

Whole bank 2 bits		**16 channels** 6 bits		**trig** 2 bits	
B1	B0	active	Chan [4:0]	T1	T0

0x605A	trig_source_2 ***(SCP only)***	16	RW	0	When above register trig_source == 0, this register allows to set any number of individual channels creating the trigger. Bit 0 corresponds to channel 0, bit 15 to channel 15. A "1" marks the channels as trigger source.
0x605C	first_hit	1	RW	1	1 = only transmit first hit 0 = transmit all hits in the window
0x605E	trigger_output	10	RW	0x100	**Defines the trigger which creates the output trigger**.This can be: one or both of the trigger inputs, lower 16 channels(B0), upper 16 channels(B1), or both banks (all channels).

Whole bank		**16 channels**		**00**	
B1	B0	active	Chan [4:0]	0	0

Bank trigger source example

Trigger 0 starts the window: bank0_trig_source = b00 0000 00 01
Channel 3 starts window: (bit 7 enables channel trigger) bank0_trig_source = b00 1000 11 00
Whole bank 0 may start the window: bank0_trig_source = b01 0000 00 00

When whole bank is selected , the channel creating the trigger is the first to trigger, but random within 12.5ns.

IO 0x6060	**Inputs, outputs**				
0x6060	ECL3	8	RW	0x00	INPUT lower 4 bit: 0= Off, 1= Trig0 in higher 4 bit: 0 = terminated, 1= unterminated
0x6062	ECL2	8	RW	0x00	INPUT lower 4 bit: 0= Off, 1= Sync_in, 2= Trig1 in higher 4 bit: 0 = terminated, 1= unterminated **when sync is selected also set reg 0x6096 !!**
0x6064	ECL1	8	RW	0x00	INPUT lower 4 bit: 0= Off, 1= Reset_in higher 4 bit: 0 = terminated, 1= unterminated
0x6066	ECL0	4	RW	0	OUTPUT 0 = Off, 4 = Busy, 8 = data in buffer above threshold 0x6018 (= Data ready) 9 = events in buffer above threshold 0x601E
0x6068	NIM4	2	RW	1	INPUT 0 = Off, 1= Trig0_in
0x606A	NIM3	2	RW	1	0, 1 = Off, 2 = Sync in **when sync is selected also set reg 0x6096 !!**
0x606C	NIM2	2	RW	1	0 = Off, and 1 = Trig1_in, 2 = Reset
0x606E	NIM1				always Trig_out

0x6070	**Test pulser**				
0x6070	pulser_status;	1	RW	0	0 = Off, 1 = On (fixed frequency of about 1.6kHz) Degrades the input signals, so only use for test.
0x6072	pulser_amplitude	12	RW	400	maximum amplitude: 0xFFF = 4095 Max value corresponds to about 30% at gain=1. Gain jumpers are situated before pulser coupling, so have no effect on the pulser amplitude.
0x6074	NIM0	4	RW	1	0= Off, 1= Cbus, 4 = Busy_out (= FIFO full or ACQ stopped) 8 = data in buffer above threshold 0x6018 9 = events in buffer above threshold 0x601E
0x607A	monitor_on	1	RW	0	switch monitor on
0x607C	set_mon_channel	4	RW	0	set channel to monitor
0x607E	set_wave	2	RW	0	set wave for to monitor

IO selection Overview

The numbers in the table can be written to the corresponding register.
For example: ECL2 should input the trigger 1: write 2 to 0x6062.

The **monitor outputs at NIM2 and NIM3** always override the register setting when they are activated by the front panel switch. When acquisition is started (0x603A), NIM2 and NIM3 fall back to their register values. But they can be reactivated by the front panel switches.

IO	default	TR0	TR1	SYN	RES	TRout	Busy	Rdy data	Rdy event	Cbus
ECL3	0	1								
ECL2	0		2	1						
ECL1	0				1					
ECL0	0						4	8	9	
NIM4	1	1								
NIM3	1			2						
NIM2	1		1		2					
NIM1	1					1				
NIM0	1						4	8	9	1

Selection of 0 always means it is unused or "Off"

Description

TR0 = Trigger 0 input
TR1 = Trigger 1 input
SYN = external Frequency to synchronize event time stamp
RES = reset for event time stamp
TRout = trigger output
Mon0 = analog output to monitor internal filtered signals, noise, Pole zero
Mon1 = analog output to monitor internal filtered signals, noise, Pole zero
Busy = module not ready to take more triggers
Rdy dat = data in buffer above threshold register "0x6018"
Rdy eve = events in buffer above threshold register "0x601E"
Cbus = control bus to control external mesytec modules (MHV-4, MPRB-32...)

Mesytec control bus:

MRC 0x6080	Module RC				
0x6080	rc_busno	2	RW	0	0 is external bus, comes out at busy output
0x6082	rc_modnum	4	RW	0	0...15 (module ID set with hex coder at external module)
0x6084	rc_opcode	7	RW		3 = RC_on, 4 = RC_off, 6 = read_id, 16 = write_data, 18 = read_data
0x6086	rc_adr	8	RW		module internal address, see box below
0x6088	rc_dat	16	RW		data (send or receive),write starts sending
0x608A	send return status	4	R		bit0 = active bit1 = address collision bit2 = no response from bus (no valid address)

Send time is 400 us. Wait that fixed time before reading response or sending new data.
Also polling at 0x608A for bit0 = 0 is possible
The Trigger0-LED shows data traffic on the bus, the Trigger1-LED shows bus errors
(i.e. non terminated lines)

Example for controlling external modules with mesytec RC-bus:

Initialize and read out a MCFD16 CFD- module.
MCFD16 ID-coder set to 7
Bus line must be terminated at the far end.

Activate MDPP-16 control bus at busy line

Write(16) addr 0x6074 data 1

Get Module ID-Code (=Type of module = 26 for MCFD16)

```
Write(16)    addr 0x6082  data 7     // address module 7
Write(16)    addr 0x6084  data 6     // send code "read IDC"
Write(16)    addr 0x6088  data 0     // initialize send request. Data has no effect
```

Wait loop: Read(16) 0x608A and compare bit0 to get 0. Then evaluate other bits for error status

```
Read(16)     addr 0x6088  data 40    // at ID readout the bit 0 shows the module RC status
                                     // (1 is on). Bit 1..7 show the IDC
                                     // → interpretation: Module off, IDC = 20
```

Set threshold for channel 0 to 10

```
Write(16)    addr 0x6082  data 7     // address module 7
Write(16)    addr 0x6084  data 16    // code "write_data"
Write(16)    addr 0x6086  data 0     // address module memory location 1
Write(16)    addr 0x6088  data 10    // start send . Data to send
```

Wait loop: Read(16) 0x608A and compare bit0 to get 0. Then evaluate other bits for error status

Optional the read back data is available.

```
Read(16)     addr 0x6088  data 10    // read back written data for control
```

Read threshold of channel 0

Write(16) addr 0x6082 data 7 // address module 7
Write(16) addr 0x6084 data 18 // code “read_data”
Write(16) addr 0x6086 data 0 // address module memory location 1
Write(16) addr 0x6088 data 0 // send read request. Data has no effect

Wait loop: Read(16) 0x608A and compare bit0 to get 0. Then evaluate other bits for error status

Read(16) addr 0x6088 data 10 // read out data, “10” returned

Activate RC in module

All set data will get active. This can also be done before setting the values.

Write(16) addr 0x6082 data 7 // address module 7
Write(16) addr 0x6084 data 3 // send code “RC_on”
Write(16) addr 0x6088 data 0 // initialize send request. Data has no effect

Deactivate MDPP-16 control bus at busy line

Write(16) addr 0x606E data 0 // busy output used as busy

CTRA

Time stamp counters, event counters
All counters have to be read in the order: low word then high word !!!
They are latched at low word read. The event counter counts events which are written to the buffer.

CTRA 0x6090	counters A				
0x6090	Reset_ctr_ab	2	RW		b0001 resets all counters in CTRA, b0010 resets all counters in CTRB, b1100 allows single shot reset for CTRA with first edge of external reset signal. the bit bx1xx is reset with this first edge Reset of "counters A" will also reset the global 46 bit TDC time stamp
0x6092	evctr_lo	16	R	0	event counter low value
0x6094	evctr_hi	16	R	0	event counter high value
0x6096	ts_sources	5	RW	b00	[ext_reset, frequency_source] bit0: frequency source (VME=0, external=1) bit1: external reset enable = 1 **QDC:** bit4: CTRB "time" counts trigger outputs (= free triggers, selected by 0x605E)
0x6098	ts_divisor	16	RW	1	time stamp = time / (ts_divisor) 0 means division by 65536
0x609C	ts_counter_lo	16	R		Time low value
0x609E	ts_counter_hi	16	R		Time high value

CTRB

Counters are latched when VME is reading the low word
Output value is divided by 40 to give a 1 us time basis

CTRB 0x60A0	counters B				
0x60A8	time_0	16	R		Time [1 us] (48 bit)
0x60AA	time_1	16	R		
0x60AC	time_2	16	R		
0x60AE	stop_ctr	2	W	0	0 = run, 1= stop counter bit 0 all counter B bit 1 time stamp counter (A)

Multiplicity filter

MULT 0x60B0					
0x60B0	high_limit0	8	RW	255	upper limit of data words in event. One responding channel may produce 1 or two data words, depending on register 0x6044.
0x60B2	low_limit0	8	RW	0	lower limit of data words in event.

Events are accepted when : low_limit <= responding channels <= high_limit;

Channel addressing

MULT 0x60B0					
0x6100	select_chan_pair	4	RW	8	channel to be modified: 0..7 channel pairs; chan 0,1 = 0, chan 2,3 = 1, ... 8 = all channels (set to common values)

User Channel setting for a channel pair, software module "SCP"
Parameters "threshold" and "PZ" are for individual channels, so two parameters per quadruple.
All channels can be set simultaneously by addrerssing 0x6100 = 8.

Address	Parameter.			default	Comment
6110	**TF_int_diff:**	7	RW	**2**	All times are in multiples of 12.5 ns. Common for 2 channels TF-integration/differentiation time, chan 0/1 valid values 1...125 (12.5 (15) ns to 1.6 us)
611A	**Gain:**	15	RW	**300**	common for 2 channels, gain x 100 gain 1...200, chan 0/1; setting 100 (gain=1) ... 19000 (gain = 200)
611C	**threshold0:**	16	RW	0xff	0 to 64k (65535) . 64 k corresponds to full range. sets lower channel (ex. chan4 when pair 2 is selected)
611E	**threshold1**	16	RW	0xff	sets upper channel (ex. chan5 when pair 2 is selected)
6120	**threshold2**	16	RW	0xff	sets upper channel (ex. chan5 when pair 2 is selected)
6122	**threshold3**	16	RW	0xff	sets upper channel (ex. chan5 when pair 2 is selected)
6124	**Shaping_time**	11	RW	**100**	common for 2 channels , FWHM-width values 4...1999 (= 50 ns to 25 us)
6126	**BLR**	2	RW	2	common for 2 channels, Base line restorer setting, 0 = off, 1 = strict (int. time = 4 shaping times), 2 = soft (int. time = 8 shaping times)
612A~	**signal_rise_time**	7	RW	**0**	common for 4 channels -default = 0, for Si-detectors, constant rise time detectors -> shortest dead time -for germanium detectors with position dependent rise time, set to largest signal rise time. This results in highest resolution and ballistic loss correction.
6112	**PZ0:**	16	RW	0xffff	signal decay_time0, lower channel (for PZ compensation) valid: 64...64k (65535), 0.8 us to 800 us, and infinite
6114	**PZ1:**	16	RW	0xffff	signal decay_time1, upper channel
6116	**PZ2:**	16	RW	0xffff	signal decay_time1, upper channel
6118	**PZ3:**	16	RW	0xffff	signal decay_time1, upper channel

How to set channel parameters

TF_int_diff is the integration and differentiation time for the timing filter.
It must not be set to a higher value than the shaping time!

1) If the timing resolution has to optimised, the integration time should be set to the rise time.
2) If a very low threshold is required, it may be necessary to set it to a larger value than the rise time, maximum value is 127 (=1.6us) or the shaping time.

PZ0 / PZ1: decay time of the pulse. The parameter must be very precise to minimise under- or overshoot of the shaped signal. The value is usually not known very precise, so it is easiest way to minimise the under- or overshoot with the monitor signals (select Tmon 3). An automatic adjust run is under consideration. Example: the decay time is 25us (time the signal needs from 100% to 36.8%) so the set value is 25000/12.5 = 2000.

Gain: The gain can be set in 1% steps. The resulting range (input voltage for highest channel in the spectrum) can be calculated as: Range = Gain_Jumper_Range / Gain.
For example: Gain-jumper with 3V label, Gain setting = 1000 -> Gain = 10, so maximum signal will be 0.3V.

threshold0 / 1: Threshold setting, 64k is full range.
The threshold is required to detect a signal out of the noise. The threshold also signals an approximated noise level to the BLR.
Example: for a low noise application the noise is 1E-3 of the full range. Setting the threshold to 3x noise this results in a set value of 64k*1E-3 * 3 = 197.

Shaping_time: is the integration time of the shaping filter. The shaping is triangular, the shaping time corresponds to the width of the pulse at half maximum (FWHM).
Compared to the traditional "shaping time" the FWHM is about a factor of 2 longer.
Example: the traditional shaping time should be 1us, the the FWHM shaping time has to be set to 2us, this results in a set value of 2000/12.5 = 160;

BLR: can be set to

0: off
1: soft, may have slight advantages for very low noise signals
2: default, compensates also for faster baseline deviations.

reset: additional time the channel is reset at an overflow. The total reset time is: shaping time + reset. The default value of 16 is the shortest possible value, and usually need not to be modified.

signal_rise_ time: Default = 0; only needed for detectors with large rise time variation, when the rise time is in the order of magnitude of the shaping time, and when high amplitude resolution is required. This parameter increases the flat top of the shaping pulse. Allows for ballistic loss correction when set to the largest possible rise time. Example: radial germanium detector with maximum 250 ns rise time. Set this parameter to 20.

Data handling

The event buffer is organized as a FIFO with a depth of 64 k x 32 bit.

Data is organized in an event structure, maximum size of one event is 255x 32-bit words
(Header, End of event, 251 data, extended time stamp, fill word).

Event structure

Word # (32 bit)	Content
0	Event header (indicates # of n following 32-bit words)
1	Data word #1
2	Data word #2
...	...
n-1	Data word #n-1
n	End of event marker

Event Header (4 byte, 32 bit)

Short #1																Short #0															
Byte #3								Byte #2								Byte #1								Byte #0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hsig		subheader						module id								tdc res		adc res		# of following words											
0	1	0	0	x	x	x	x	ii	ii	ii	ii	ii	ii	ii	ii	t	t	t	a	a	a	n	n	n	n	n	n	n	n	n	n

hsig: header signature = b01

subheader id: currently = b00xxxx → Byte #3 = 0x40

module id: depending on board coder settings → Byte #2 = Module ID

tdc res: TDC resolution, depending on register 0x6042

adc_res: ADC resolution, depending on register 0x6042

# of follow. words: indicates amount n of following 32-bit words:
n-1 events + 1 end of event marker)

Data words (4 byte, 32 bit) DATA-event

Short #1																Short #0															
Byte #3								Byte #2								Byte #1								Byte #0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dsig		fix |						|flags		T	channel #				data (16 valid bits)																
0	0	0	1	x	x	x	x	p u	o v	t	c	c	c	c	c	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d

dsig: data signature = b00

fix: bit field marking a data word = b0001xxxx → Byte #3 = 0x1x
(extended time stamp marked as = b0010xxxx)

pu: pile up detected

ov: overflow or underflow

T: T = 1: Trigger channel, {T,chan#} = 32 for trig0, or = 33 for trig1

channel #: T = 0: channel # runs from 0 to 15 for amplitudes, 16 to 31 for time.

within an event buffer, channels may occur in arbitrary order

data: conversion data, data width up to 16 valid bits

End of Event mark (4 byte, 32 bit)

Short #1																Short #0															
Byte #3								Byte #2								Byte #1								Byte #0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
esig		trigger counter / time stamp (30bit)																													
1	1	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t

esig: end of event signature = b11

trigger counter/ time stamp: 30 bit trigger counter or time stamp information, depending on register 0x6038 “marking type”: 0 = event counter, 1 = time stamp

When in single event mode (register 0x6036 = 0), reading beyond EOE, MDPP-16 emits a VME Berr (bus error).

When in multi event mode 3 (register 0x6036 = 3), reading beyond EOE after the limit specified in register 0x601A, MDPP-16 emits a VME Berr (bus error)

This can be used to terminate a block transfer or multi block transfer.

Initializing the MDPP-16 for basic measurement

The power up initialization if ever possible was chosen to allow easy start.
The following steps help to get an individual setting from the beginning.

1. Choose channel input signal. This may be differential input (only MDPP with header input connector) or unipolar input (MDPP with header or Lemo inputs).
   For differential input, choose the differential input jumpers and put them in terminated (110 Ω) or unterminated position. Connection is usually done with twisted pair cables.
   For unipolar input, use the unipolar jumpers. Depending on position they provide terminated (50 Ω) or unterminated inputs. For MDPP with header input connectors, an adapter to Lemo may be helpful (MAD 34_16 SM or SF).

2. register setup of the channels (reg 0x6100 to 0x612F)
   set reg 0x6100 to 8 (copy values to all channels)
   set reg 0x6110, tf_int_diff to the rise time of your preamp signal. For example 50ns/12.5 - set reg to 4;
   set reg 0x6112 / 0x6114 to decay time, For example 25000ns/12.5ns = 2000;
   set reg 0x611A gain. For example max input signal is 100mV,
   gain jumper 3V -> gain = 30, set reg to 3000;
   set reg 0x611C / 0x611E threshold. For example threshold should be 0.5% of max range,
   64k*0.005 = 328;
   set reg 0x6124 shaping time. For example 1us shaping time = 2us FWHM width;
   2000ns/12.5ns=160;
   registers 0x6126 to 0x612A can be left at default.

3. Choose trigger source. The trigger starts a window of interest, which can start in the past or in the future shifted by 25 us, and can have a width of 1.6 ns to 25 us. The two trigger inputs can be used, but also any channel and the whole bank. To select the trigger source, reg 0x6058, trig_source is used. In easiest case the module is self triggering. Then set the register to 0x100 (whole bank triggers) ;

4. Choose trigger timing:
   Window start, for example the signals which belong to one event arrive within 1us at the different channel inputs. The fist signal to arrive will trigger. So the window start should be at –50 ns in the past and should last for 1000 ns. So set 0x6050 win_start = 16384 – (50ns / 1.56ns) = 16352
   Set width 0x6054 win_width = (1000ns / 1.56ns) = 640.

5. For timing **Resolution**, you may choose a channel width of 100 ps (set 0x6042 = 2), so the data will fill a 10 k spectrum. The triggering channel will create a peak in timing spectrum at 50ns (chan 512).
   For amplitude **Resolution**, you may choose 8k, so set reg 0x6046 to 3.

6. See chapter "**The MDPP-16 read out"** to initialize the readout section of the MDPP.

The MDPP-16 read out in two modes

Single event readout

In this mode the data are collected within the window of interest, starting with an external trigger. The data are then stored in a memory and the module waits for the VME readout. After readout of the data at 0x0000 the register 0x6034 is written and allows a new gate to start the conversion. Gates coming within the time from fist gate to writing the 0x6034 register are ignored.
For dead time the conversion time plus latency and VME readout time add up.

1. Assumed: 32 bit read (D32 or BLT32)
   Wait for IRQ to start readout of an event
   Read register #6030 for event length
   Read from buffer event_length + 1
   Write reset register 0x6034

2. After IRQ, start block transfer until BERR on VME-bus
   Then write reset register 0x6034

Example

Stop acquisition: start_acq 0x603A = 0; Stop
Set multi event register 0x6036 = 0 (default).
At power up reset or after soft reset, the IRQ register is set to 0 (no interrupt)

Initialize IRQ (for example to IRQ1, Vector = 0):
set IRQ:
set reg 0x6012 to 0 (IRQ Vector)
set reg 0x6010 to 1 (IRQ-1 will be set when event is converted)

Reset FIFO: write register 0x6034 (any value)
start_acq: 0x603A = 1; Start

Now module is ready for IRQ triggered readout loop:
→ IRQ
Read register 0x6030 for event length (D16)
Read from buffer event_length + 1 (BLT32)
Write reset register 0x6034 (D16)
Or:
→ IRQ
Start block transfer (BLT32) until BERR on VME-bus
Then write reset register 0x6034 (D16)
The above procedure works completely unchanged **with multi event mode 0x6036 = 3 and 0x601A = 0**. In this mode the buffer is used but the data are read out event by event.
After each event a Berr is emitted, which is removed by writing the 0x6034 readout reset.

Multi event readout

In multi event readout mode (0x6036, multi_event = 1 or 3) the input is decoupled from output by an 48 k words buffer. So the input is ready for a new trigger after trigger dead time.

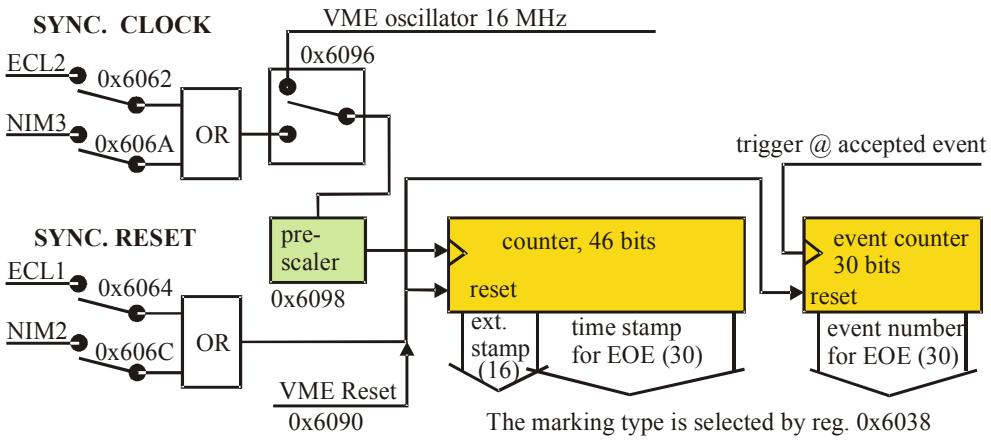
When several converter modules are used in one setup, there has to be a way to identify coincident

data from different modules which belong to the same event.

Event synchronization

One method is **event counting**.
Each module has an event counter and counts the incoming gates. In complex setups, the gates are best initiated by the individual detector timing signals and significant amount of logic and timing modules have to be established and adjusted to coordinate the detector triggers. A single timing error in all the experiment run time, which will allow an additional gate to come to some module or a suppression of a gate, will corrupt the complete data set, as data gets asynchronous.

The better one is **time stamping.**
A central synchronization clock (sync) (for MDPP-16 this can be the VME built in clock of 16 MHz or an external clock up to 75 MHz) is counted to create a time basis. At experiment start the time counters of all modules are reset via a VME multicast write instruction to reset register 0x6090, or by an external reset signal.
All incoming events are then labeled with a 30 bit long time tag (when extended time stamp is set, an extra 16 bits are added). At data analysis the data streams from different modules are analyze and correlated events are grouped for further processing.



The graphics shows how event counter and time stamper are operated. At high sync clock frequencies, the 30 bit time stamp may overflow. In this case another 16 bit of time stamp can be added to the data.

The synchronization methods allow the different modules to be completely independent from each other. It gets now possible to use large data buffers in the front end modules, and do the readout when the VME data bus is not occupied. The MDPP-16 allows to set a buffer fill threshold which emits an interrupt when the data fill level in the buffer exceeds the threshold.

Data transfer

In principle any amount of data can be read at any time from the buffer, but then events may be splitted to two consecutive readout cycles, which normally is no problem.
When only full events should be read in one readout cycle, there are two possibilities.

1. multi event mode = 1: read “buffer_data_length” (0x6030) and transfer the amount of data read there.
2. multi event mode = 1: The buffer must be read to the end which means to the Berr mark. Note that this in principle requires to read an infinite number of words, as the conversion can produce more data than can

be read via VME-bus.

3. So if high rates can appear, the data acquisition should at least be tolerant to splitted events.
   an easier way to overcome those problems is to use multi event mode = 3 and limit the data transfer via register 0x601A to a reasonable amount (for example 1000 Words). A “Berr” is then emitted after the next “EOE” marker exceeding the word limit. After readout, 0x6034 has to be written to allow transmission of a new data block.

IRQ

For many setups it is useful to control the readout via interrupt requests (IRQ) defined by VME.
For MDPP-16 an IRQ is initiated when the buffer fill level gets above the “irq_threshold” (0x6018).The IRQ is acknowledged by the VME controller, then the controller starts a readout sequence. When not using the readout reset (0x6034) at the end of a readout cycle, the MDPP does not know when the cycle ends. The IRQ is then set again when the data fill level exceeds the irq-threshold. When not enough data are read from FIFO to drive the FIFO fill level below the threshold, no new IRQ will be emitted.
So for a readout which is stable against any external influences (readout delays, high input rates),
we recommend to write the readout_reset after each readout sequence. For several mesytec modules in a VME bin, this can also be done with a single multicast write.

Example 1, multi event readout

1. **Stop acquisition**
   start_acq 0x603A = 0; Stop

2. **Time stamping**
   The module will use here an external reference synchronization clock and will be reset (synchronized) via VME command.

Set synchronization clock input	ECL2	0x6062 = 1;
Set sync clock source, reset source	ts_sources	0x6096 = 1; (ext osc, int reset only)
Show time stamp in EOE mark	marking type	0x6038 = 1;
Synchronization:	Reset_ctr_ab	0x6090 = 3; reset all counters

3. **IRQ**
   Initialize IRQ (for example to IRQ1, Vector = 0):
   set IRQ:
   set reg 0x6012 to 0 (IRQ Vector)
   set reg 0x6010 to 1 (IRQ-1 will be set when event is converted)
   set reg 0x6018 to 200 (IRQ emitted when more than 200 words in FIFO)

4. **Set Multi event**
   Multi event 0x6036 = 3 multi event with limited data transfer
   Max_transfer_dat 0x601A = 200 transmit maximum 200 words + rest of event before sending Berr

5. **Buffer initialization, start**
   FIFO_reset 0x603C = 0;
   Readout reset 0x6034 = 0;
   start_acq 0x603A = 1; Start

6. Readout loop

→ IRQ

Start multi block transfer (BLT32) until BERR on VME-bus
Then write reset register 0x6034 (D16)

Example 2, chained block transfer

Describes multi event readout but with 3 MDPPs and chained block transfer

To operate several modules in one VME bin, each module has to be given a different address.
The 4 coders on the main board code for the highest 16 bits of the 32 bit address. Best way is, to use only the highest 8 bits for coding (2 rotary coder marked with high). It makes sense to use the slot number as high address. So:

MDPP1 in slot 1 gets 0x0100
MDPP2 in slot 2 gets 0x0200
MDPP3 in slot 3 gets 0x0300

If you don't change the module ID default, the modules will now also have the ID 1…3 which will be transmitted in the data header.

Now initialize the individual modules:

MDPP1: set 0x0100 6020 to 0xA2 (CBLT first module, Multicast enable)
MDPP2: set 0x0200 6020 to 0x82 (CBLT mid module, Multicast enable) also any further module in the middle of the readout chain is initialized this way.
MDPP3: set 0x0300 6020 to 0x8A (CBLT last module, Multicast enable)
When you don't change the default addresses for CBLT and MCST, the modules will have the CBLT start address of 0xAA00 0000 and the MCST start address of 0xBB00 0000.

You can now do the initialization 1) to 5) of Example 1 via multicast at the offset address 0xBB00.

The readout loop has to be modified slightly:
→ IRQ
Start multi block transfer (BLT32, MBLT64) at address 0xAA00 0000 until BERR on VME-bus
Then write reset register 0xBB00 6034 (D16) at the multicast address.

Note: use multi event mode 0 or 3 for CBLT (mode 1 will not work !)

Special VME Operation

MBLT64
MBLT64 is defined by the address modifier. The word alignment within the transmitted 64 bit words is kept by adding fill words at odd word numbers.

CMBLT64
Is intrinsic when chained block transfer is used with MBLT64.

Dead time

the present SCP firmware needs 600 ns for calculations and event building. This dead time has only effect when the shaping time (FWHM) is less , otherwise (starting from 600 ns shaping time) the device adds no dead time.
Another aspect is the window of interest. While it is open, no further window can be opened, and some processing time is required. The trigger dead time is window width + 900 ns.

Trigger output delay.

The trigger output delay measure from rising edge of the input signal is 400 ns + the timing filter integration time.

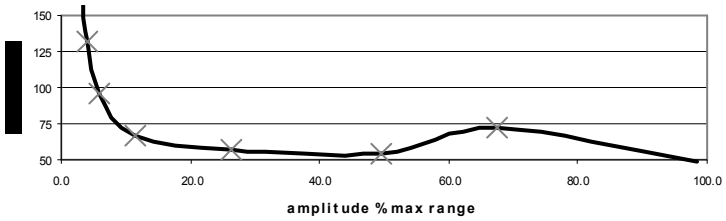
Detailed measured data from software module "SCP"

Amplitude resolution

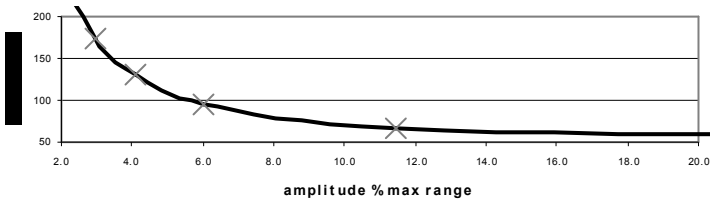
Due to low input noise, The MDPP-32 allows very high amplitude resolutions.
The input noise is slightly higher (Factor 1.3) than for MDPP-16. This may have an effect at very low input ranges. Detailed measurement will follow.

Timing resolution

MDPP-32 provides a very good timing resolution of 75 ps rms for the time difference between two channels. The resolution of time difference between input trigger and one channel is less than 75 ps rms in the amplitude range of 10% to 100% .
(Measurement condition: maximum range = 2 V, rise time 20 ns, TF integration 25 ns)



Timing resolution, trigger input starts, one channel stops.
Amplitude of the channel input from 3% to 100% of full range.

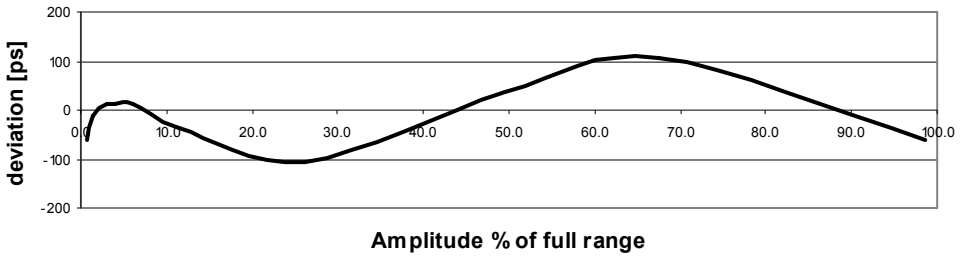


As upper figure, zoomed to lower 20% .

In contrast to other approaches, the timing resolution is independent of the signal delay between channels.

Timing walk (time drift with amplitude)

The MDPP-32 has implemented a numerical constant fraction discriminator, so the timing walk is mostly eliminated. From 3 % to 100 % of full amplitude range, the walk is ± **110 ps**.
(Measurement condition: maximum range = 2 V, rise time 20 ns, TF integration 25 ns)



Timing walk, trigger input starts, one channel stops

The amplitude of the channel input is increased from 3% to 100% full range.

Further Software Modules:

Self timed QDC and TDC to analyze fast pulses, pulse shape analysis
- Amplitude resolution 4 k
- Timing resolution 70 ps rms trigger to channel

Planned:
Peak sensing ADC
- No dead time beyond input pulse width.
- 16 k resolution
- low INL and DNL