mesytec **MTDC-32** is a fast high resolution time digitizer. It is internally realised as a 32 + 2 channel time stamper (32 channels + 2 triggers). It can be operated like a traditional TDC with a start signal (at trigger inputs) and 32 stop signals, so needs no further synchronisation. Due to the internal time stamper design, the start trigger can be shifted in time by ±16 us, so no external delays are needed, even when a delayed experiment trigger is used as start signal. A pure time stamper mode allows to output all 34 digitized input signals independently with 46 bit time stamp.

Features:

- 32 + 2 channel time stamping TDC
- Channel to channel or trigger to channel TOF resolution
- typical 5 ps maximum **10 ps** rms
- Conversion time 160 ns
- Two operation modes:
  - start-stop mode with configurable window of interest and 16 bit conversion output
  - time stamper with 46 bit time stamp
- High quality conversion with very low INL and DNL
- 48 k (32 bit-) words multi event buffer
  (1 word corresponds to 1 converted channel)

- Channel inputs configured by Jumpers:
  - differential: ECL, LVDS and LVPECL, terminated or unterminated
  - unipolar: register configurable threshold, NIM, TTL or analogue signals possible.

- Supports different types of event synchronisation stamping (based on VME-clock or external clock)
- Independent bank operation
- 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
- Live insertion (can be inserted in a running crate)
- Power consumption: 22 W, +5 V and ±12 V needed

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

Overview

MTDC-32 is a fast 32 + 2 channels high resolution time digitizer. It is internally realised as a 32 + 2 channel time stamper.

Trigger inputs
Two extra channel inputs (trigger inputs) are available. They can be used (as any other channel) to start a window of interest. Width and delay can be set in steps of 1 ns. The two extra trigger inputs help to preserve the usual symmetry of detector channels, which are power of 2.
If the two banks are operated independently, any of the 16 channels of one bank or the corresponding trigger channel can be selected as window start input.

The MTDC works as easy as traditional TDCs, the synchronisation between several modules is simply done by a central trigger which performs start / stop with highest resolution
(for MTDC-32 it can also be shifted in time to avoid external delays).

From trigger input a window is generated which can be delayed from -16 us (this means data delayed by 16□s) and +16 us in steps of 1ns. The width of the window of interest can be up to 16 us, but maximum 64 k x channel width (for a resolution of 4ps this means maximum window is 256 ns). The channel width is defined by the resolution register. The time per histogram channel
(channel width) can be set by resolution register between 4 ps and 250 ps.
All hits registered within the window of interest are written into one event structure. A 16 bit value is stored in the event structure, which is the difference of window start and the hit time.
It is possible to choose if only the first hit of one channel detected within the window of interest will be registered, or all hits. The total event structure is limited to 240 hits.

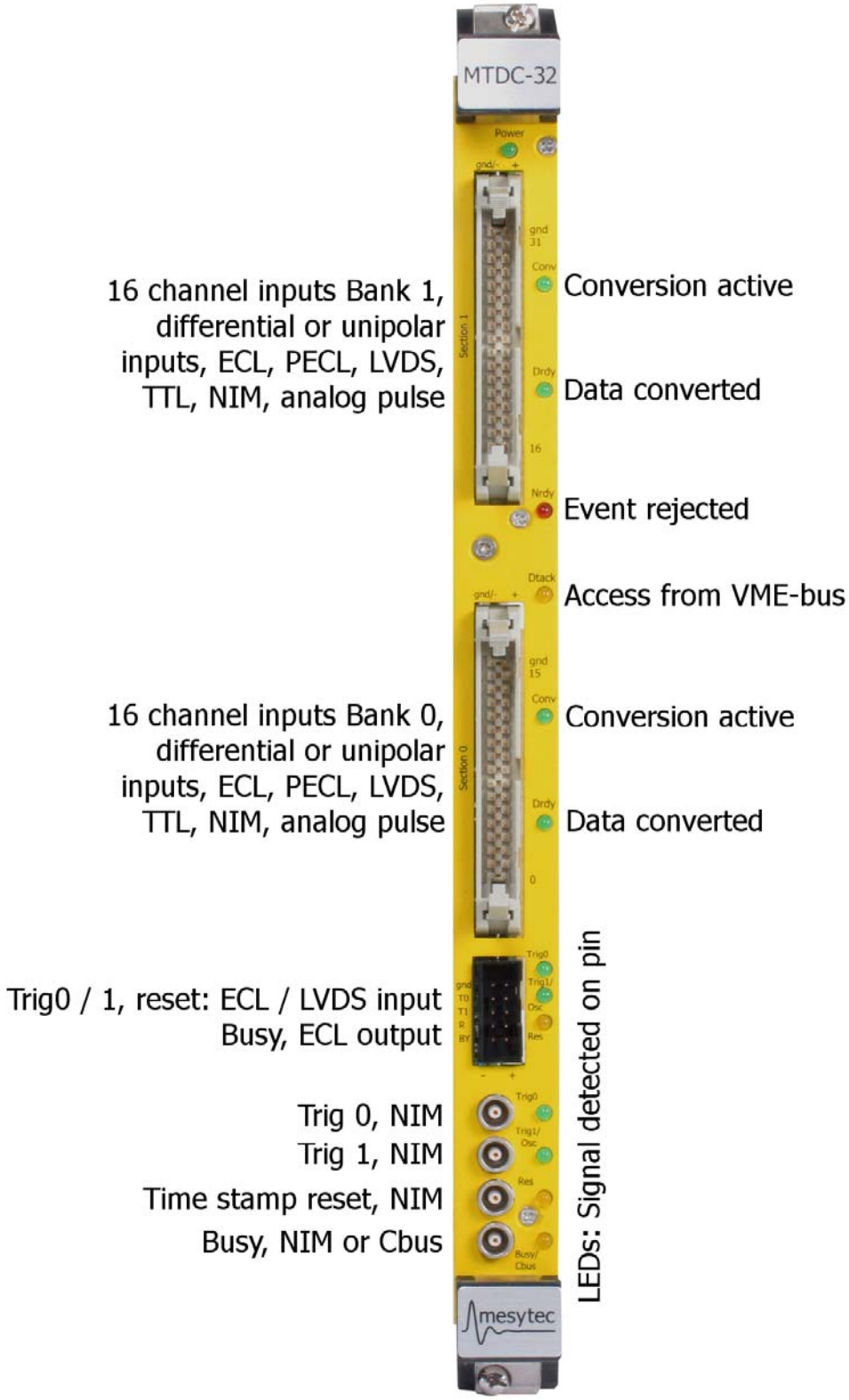
For special applications a **pure time stamper mode** is available. That's a mode which does not use a window of interest and does not calculate a time difference, but transmits each hit with a 46 bit time stamp (runs 281 s). Each hit is transmitted as an independent event.

A built in Multiplicity filter helps to reduce data by skipping events with unwanted numbers of hits. A lower and an upper limit can be defined.

The MTDC-32 as all mesytec VME modules provides event time stamping.
The event time stamping has nothing to do with the high resolution time stamping of the TDC !
It is only a rough time tag for event structures, to synchronise simultaneously converted events in different modules. Time basis is an external oscillator or the VME clock (16 MHz). The feature is identical to the other mesytec VME-modules: MADC-32, MQDC-32, MTDC-32, MDI-2.
It is used to tag event data arriving at the same time at different digitizer modules. This is relevant when several modules are operated asynchronously. The Asynchronous readout of modules allows to take advantage of the large FIFO buffers implemented in the modules. The VME-data readout timing is then decoupled of the input data stream.

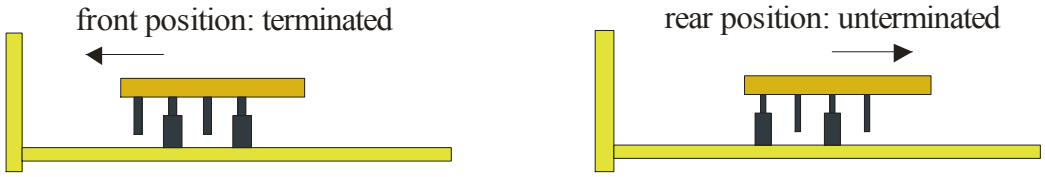
The data structure of MTDC-32 is identical to the other mesytec modules. Also all data handling registers are identical.

MTDC-32 front panel

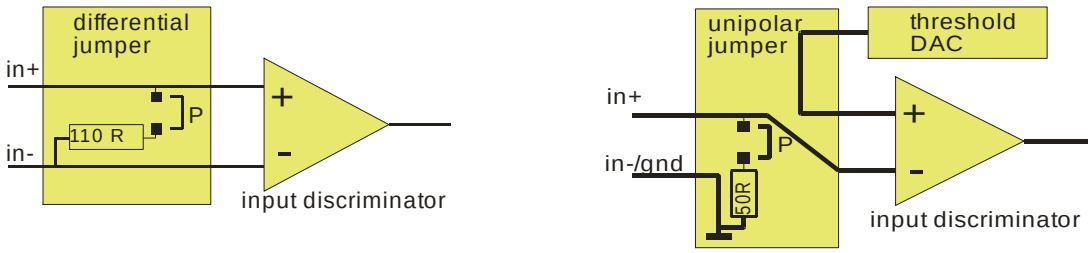


Channel inputs Bank 0 and 1

MTDC-32 has very flexible inputs, which can be configured by the two sets of input jumpers (unipolar and differential), and by setting a register configurable threshold
(only with unipolar jumpers). Both jumper sets are included at delivery.
The Input termination is coded by the jumper position:



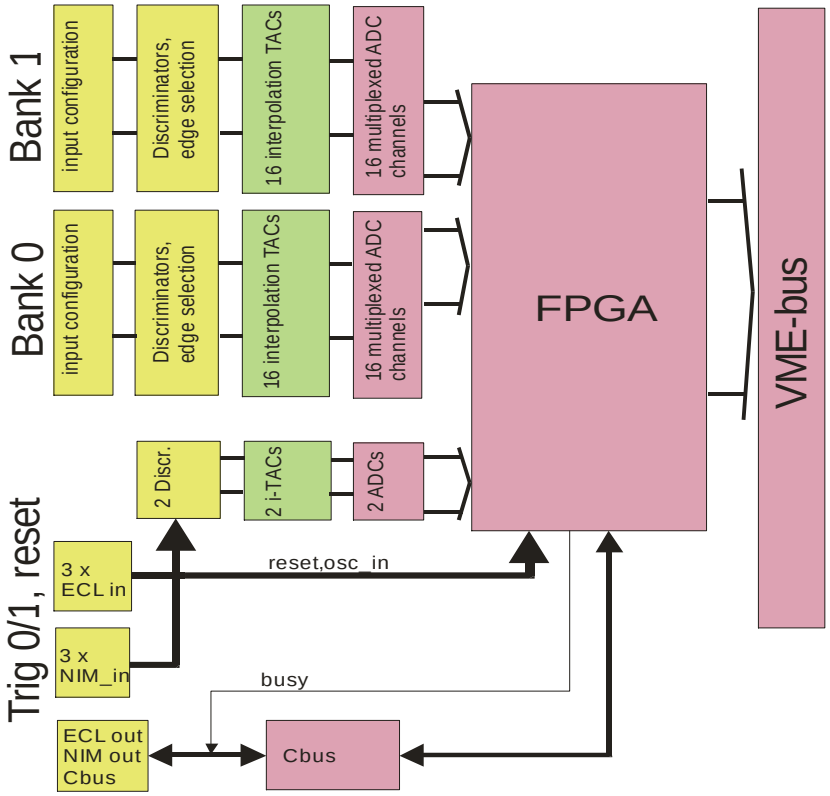
The differential jumper set allows for differential input signals (ECL, LVPECL, LVDS).
The unipolar jumper set allows for unipolar signals (NIM, TTL, analogue positive or negative signals).
A threshold of □1.5 V can be set in 256 steps. The polarity is chosen to allow NIM triggering at the leading edge with "positive edge" setting (default). For TTL signals "neg edge" should be selected.



"P" means jumper position near to front panel, then input is terminated.

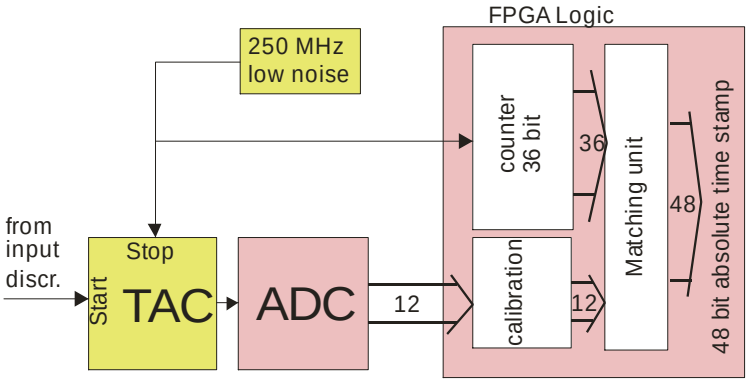
Triggering leading or trailing edge can be chosen for each bank by register setting.
This also allows to measure pulse width for 16 signals, when the 16 signals are fed to the two bank inputs in parallel. The one bank is configured to positive edge triggering, the other one to negative edge triggering.

MTDC-32 overview schematic



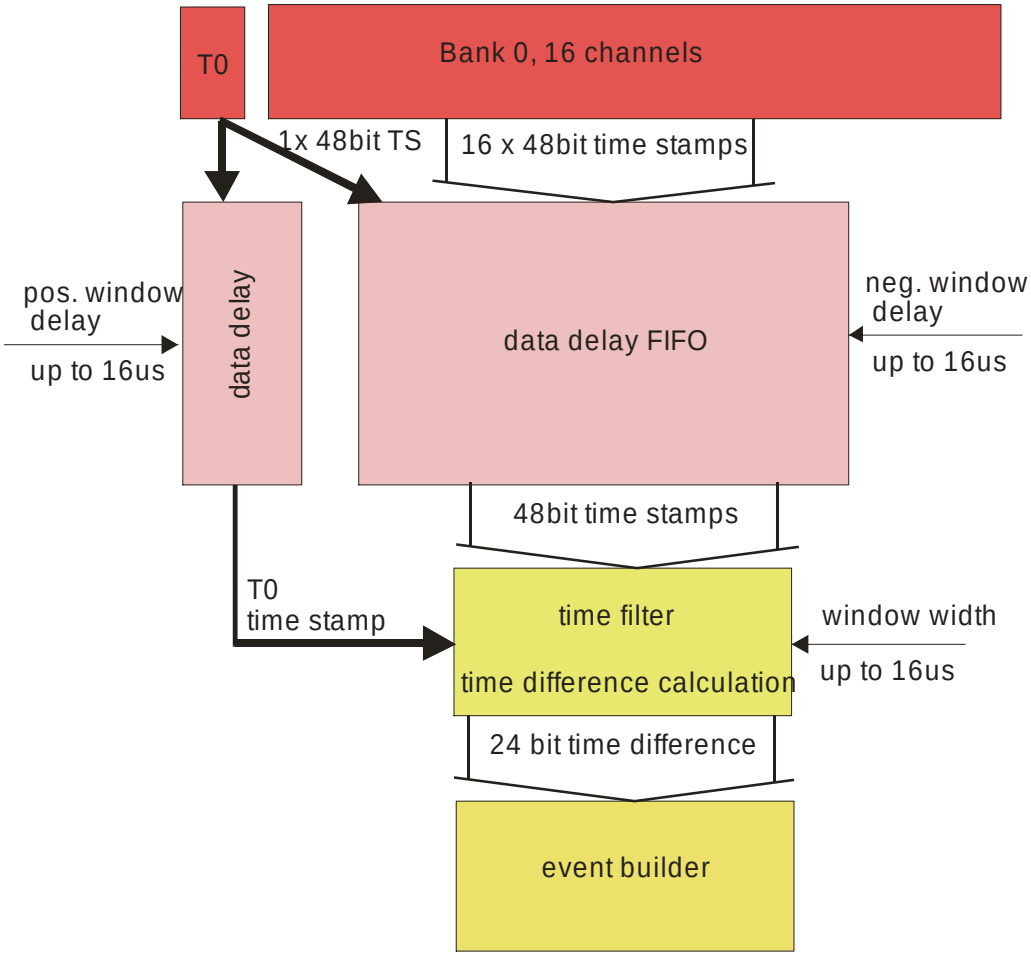
Schematic of one conversion channel

MTDC-32 is based on 34 free running interpolation time to analogue converters (TAC) with a bit resolution of 1 ps. The interpolation interval is 4 ns. Longer times are measured by counting the number of intervals with a 48 bit counter. In the matching unit the time within the interval has to be added to the correct counter value. Each channel operates completely independent and delivers a 48 bit time stamp (running 281 s).



Filtering the data by window of interest

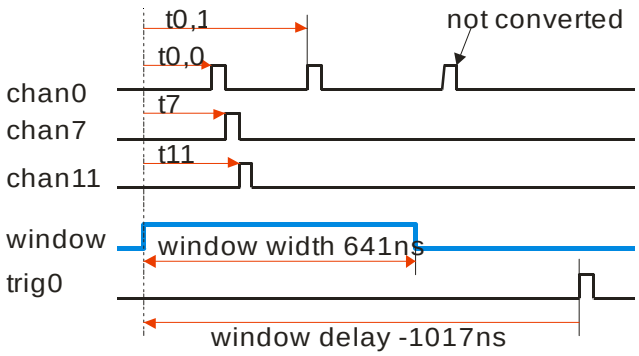
Example: only one bank is used, trigger 0 input is selected to generate the window of interest.



If the trigger input is delayed to the data = neg. delay (this is the standard case), the data have to be delayed (stored) until the trigger arrives. When the trigger arrives, the filter is opened for the time defined by "window width". The time difference between trigger time and the hit times is calculated (24 bit at 1 ps resolution have a maximum value of 16 us). When the window closes, the data are formatter in the event builder and stored in the main FIFO. When the data are delayed to the trigger = pos. delay, data pass without delay and the trigger is delayed by the "window delay".
Then filtering runs as before. In time stamping mode, delays and filter are bypassed.

The following example shows how the hit data are filtered:

A window with delay of -1017 ns and a width of 641 ns is configured. More than one hit per channel is selected.



The trig0 -1017 ns is the virtual start for the data. Channel resolution is selected to 1/64 ns (15.6 ps)
The generated event structure looks as follows (channels are in arbitrary order):

1 x header

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				0	0	# of following words									
01		0						0								4				0	0	5									

4 x data

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									channel #					TDC data (16 valid bits)															
0	0	0	0	0	1	0	0	0	0	0	0					9792															
0	0	0	0	0	1	0	0	0	0	0	0					19440															
0	0	0	0	0	1	0	0	0	0	0	7					11376															
0	0	0	0	0	1	0	0	0	0	0	11					13344															

1 x end of event

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 / synchronisation time stamp (30 bit)																													
1	1	12346890																													

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

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

Input /output	direction	termination	Default functionality	Alternate functionalities
ECL3	Input	R*	Trigger0	-
ECL2	Input	R	Trigger1	Time stamp oscillator input
ECL1	Input	R	Reset time stamp counter	-
ECL0	Output	100 R	Busy	data_threshold, event_threshold
NIM3	Input	50 R	Trigger0	-
NIM2	Input	50 R	Trigger1	Time stamp oscillator input
NIM1	Input	50 R	Reset time stamp counter	-
NIM0	Input / Output	50 R	Busy	Mesytec control bus I/O, data_threshold, event_threshold

"R" means register selectable termination.

Front Panel LEDs:

LED "Conv" digitization in progress
LED "Drdy" Data are ready converted and can be read out
LED "Nrdy" Trigger detected, but TDC busy or FIFO full. Event will be lost
LED "Dtack" Access from VME bus accepted

Lemo and differential inputs
Minimum trigger width for individual inputs is = 5 ns
Maximum external reference oscillator frequency: 75 MHz

Technical data:

Resolution @ ECL signals, 500 ps rise and fall time, time difference 10 ns	Typ. 5 ps rms, max 10 ps rms
INL	<5 ps
DNL	1 %
Crosstalk 32 channels:	5 ps max*
Crosstalk 2 triggers	10 ps max*
Time offset matching	±100 ps
Conversion, busy time	160 ns, independent for each channel 120 ns for the 2 trigger channels
Differential input range:	–2 V to +3 V, ECL, LVDS, LVPECL
Differential termination	configurable: open or 110 R
unipolar input range:	±5 V (voltages beyond this may destroy the channel)
unipolar termination	configurable: open or 50 R
Threshold for unipolar input	Configurable: 0 V to ±1.5 V, in ±127 steps
Standard Quartz Precision Stability	50 ppm (5*E-5) ageing 5ppm / year
With oven stabilized precision Quartz Precision Stability	±2 ppm trimmable 0.7 ppm first year, 4 ppm 10 years
Power consumption	(Total: 22 W) +5 V +2.2 A +12 V +350 mA –12 V –530 mA

* Due to significant crosstalk of twisted pair cables, the direct edge cross talk with 1 m of cable and signals with 500 ps rise time, is around ±30 ps for neighbour channels, but negligible for others.

MTDC32 register set

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 will be a fill word (= 0).
FIFO size: 48 k - 512 = 48640 words with 32 bit length

Header (4 byte)

2 header signature	6 subheader	8 module id	4 tdc_resolution → 0x6042	12 number of following data words, including EOE
b01	b000000	module id	bxxxx	number of 32 bit data words

Data (4 byte) DATA event

2 data-sig	8	1	5	16
b00	00 0100 00	Trigger Flag	channel number	TDC time difference

channel numbers may come in arbitrary order

Data (4 byte) Extended time stamp

2 data-sig	9	5	16
b00	00 0100 100	0 0000	16 high bits of time stamp

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

2 data-sig	9	5	16
b00	0	0	0

End of Event mark (4 byte)

2	30
b11	trigger counter / time stamp

Trigger flag is the address extension to "channel_number" to address the two trigger inputs.
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.

When in time stamper mode (reg 0x6044 set to 1) the readout data are modified the following way:

Header (4 byte)

2 header signature	6 subheader	8 module id	4 tdc_resolution → 0x6042	12 number of following data words, including EOE
b01	b000000	module id	bxxxx (no effect on data!)	number of 32 bit data words

Data (4 byte) DATA event

2 data-sig	8	1	5	16
b00	00 0100 00	Trigger Flag	channel number	16 low bits of full time stamp lowest bit: 4ps

Data (4 byte) Extended time stamp

2 data-sig	9	5	16
b00	00 0100 100	0 0000	16 low bits of event time stamp (has nothing to do with full time stamp!)

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

2 data-sig	9	5	16
b00	0	0	0

End of Event mark (4 byte)

2	30
b11	30 high bits of full time stamp

Trigger flag is the address extension to "channel_number" to address the two trigger inputs.
So the full channel address has 6bits, and runs from 0 to 33. The addresses 32 is for trigger input 0, the address 33 for trigger input 1.

For each of the 34 detected triggers 4 words are transmitted.
The events may not be in correct time order in a scale of 128 ns.

Extended time stamp is only sent when "marking type" reg 0x6038 is set to b11.
46 bits of the extended time stamp are transmitted, based on 4 ps timing resolution. It runs 281 s.
If longer time tags are required, the event time stamp may be counted up by an external clock, running for example with 1 Hz.

This will give a long term raw timing, and allow to properly increment a counter in Evaluation software to extend the full time stamp to any value.

Registers, Starting at address x6000 (access D16)

Blue marking is implemented starting with firmware FW0200

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	W		breaks all activities, sets critical parameters to default
0x600E	firmware_revision	16	R		0x0110

	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 = number of events.
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 multivalent = 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 MTDC 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, wich 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 budder is in a well defined status.
0x603C	FIFO_reset		W		Initialise FIFO
0x603E	data_ready	1	R		1 → data available

	operation mode				
0x6040	bank_operation	1	RW	b0	0 → banks connected 1 → operate banks independent
0x6042	tdc_resolution	5	RW	3	9 → 500 ps 8 → 250 ps 7 → 125 ps 6 → 62.5 ps 1 ns / 16 5 → 31.3 ps 1 ns / 32 4 → 15.6 ps 1 ns / 64 3 → 7.8 ps 1 ns / 128 2 → 3.9 ps 1 ns / 256
0x6044	output_format	1	RW	0	0 = standard (time difference) 1 = single hit full time stamp

	Trigger				
0x6050	bank0_win_start	15	RW	16k-16	Unit: ns Start window of interest from trigger start, Offset +16 k = 16384; 16k → no delay < 16 k, window starts before Trigger > 16 k, window is delayed
0x6052	bank1_win_start	15	RW	16k-16	same as for bank 0
0x6054	bank0_win_width	14	RW	32	Unit: ns, max 16 k = 16 us, unsigned
0x6056	bank1_win_width	14	RW	32	Unit: ns
0x6058	bank0_trig_source	10		1	at joined bank for common event, at split banks for bank0. **Defines the trigger which creates the window of interest**. This can be: one or both of the trigger inputs, any of the 32 channel inputs, a whole bank.
0x605A	bank1_trig_source	10		2	only at split bank, bank 1. {bank_trig[1:0], chan_trig[5:0],trig[1:0]}
0x605C	first_hit	2	RW	b11	bit0 = bank0, bit1 = bank1 1 = only transmit first hit

Whole bank		**32 channels**		**trig**	
B1	B0	active	Chan [4:0]	T1	T0

					0 = transmit all hits in the window

Bank trigger source example

Trigger 0 should start 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 more than one trigger source is selected , the channel creating the trigger is more or less random.

IO 0x6060	Inputs, outputs				
0x6060	Negative_edge	2	RW	b00	bank[1:0] flag, for differential input jumper: 1 = channel trigger on negative edge unipolar input jumper: 1 = trigger on rising edge of the signal
0x6062	ECL_term	3	RW	b000	switch ECL/LVDS terminators on (1 = on) bit 0 for: "trig0", (top ECL3 input) bit 1 for "trig1", (ECL2 input) bit 2 for "Res", (ECL1) (reset input) Unconnected inputs will be in a well defined state by internal weak pull down resistors.
0x6064	ECL_trig1_osc (ECL2)	1	RW	0	(ECL2 input) 0 → trig1 input, 1 → oscillator input (**also set 0x6096 !!**)
0x6066	ECL_out_config (ECL0)	4	RW	0	(Lowest ECL0, output) 0 → as busy (= FIFO full or ACQ stopped) 8 → data in buffer above threshold 0x6018 9 → Number of events in buffer above threshold 0x601E
0x6068	Trig_select	1	RW	0	0 → Trigger 0 and 1 from NIM-inputs, 1 → Trigger 0 and 1 from ECL-inputs
0x606A	NIM_trig1_osc (NIM1)	2	RW	0	0 → trig1 input, 1 → oscillator input (**also set 0x6096 !!**)
0x606E	NIM_busy (NIM0)	4	RW	0	0 → as busy (= FIFO full or ACQ stopped) 3 → as Cbus output 8 → data in buffer above threshold 0x6018 9 → Number of events in buffer above threshold 0x601E

0x6070	Test pulser / Threshold				
0x6070	pulser_status;	1	RW	0	0 = off, 3 = on: always trig0 and one channel from 0 to31 then trig 1, cycling. Time difference from trig0 to channel about 0s +- 2ns
0x6078	bank0_input_thr	8	RW	105	Discriminator level for unipolar input bank 0, For NIM 105 is optimum.
0x607A	bank1_input_thr	8	RW	105	Discriminator level bank1

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:

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

Activate MTDC32 control bus at busy line

Write(16) addr 0x606E data 3

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     // initialise 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     // initialise send request. Data has no effect
```

Deactivate MTDC32 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 precision 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	2	RW	b00	bit0: frequency source (VME=0, external=1) bit1: external reset enable = 1
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	RW	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	Bank0 (or B0, B1 when connected) upper limit of responding channels
0x60B2	low_limit0	8	RW	0	Bank0 (or B0, B1 when connected) lower limit of responding channels
0x60B4	high-limit1	8	RW	255	Bank 1 upper limit
0x60B6	low-limit1	8	RW	0	Bank 1 lower limit

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

Data handling

The event buffer is organised as a FIFO with a depth of 48 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				# of following words											
0	1	0	0	0	0	0	0	ii	ii	ii	ii	ii	ii	ii	ii	r	r	r	r	n	n	n	n	n	n	n	n	n	n	n	n

hsig: header signature = b01

subheader id: currently = b000000 → Byte #3 = 0x40

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

Data origin f: 1 = Data from bank 1 when banks not connected

tdc res: TDC 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								T	channel #					TDC data (16 valid bits)															
0	0	0	0	0	1	0	0	0	0	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 currently without meaning = b00010000 → Byte #3 = 0x04

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

channel #: number of TDC channel → Byte #2 = channel#

within an event buffer, TDC channels may occur in arbitrary order

TDC data: TDC conversion data, data width 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, MTDC-32 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, MTDC-32 emits a VME Berr (bus error)

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

Initialising the MTDC for basic timing measurement

The power up initialisation if ever possible is 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 or unipolar input.
   For differential input (**ECL**, LVDS...), 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. An adapter to Lemo may be helpful (MAD 34_16 SM or SF).
   When the input signal is **NIM**, the default values for threshold and edge selection are fine.
   When the input signals are **TTL**, the thresholds (0x6078 bank0_input_thr, and 0x607A bank1_input_thr) has to be set to 255, and reg 0x6060 set to 3 (= both banks on positive signal edge).

2. Choose trigger source. The trigger starts a window of interest, which can start in the past or in the future shifted by 16 us, and can have a width of 1 ns to 16 us. The trigger replaces the common start or common stop signal of traditional TDCs. The two trigger inputs can be used, but also any channel and whole banks. To select the trigger source, reg 0x6058, bank0_trig_source is used. When trigger input 0 should be used, set this register to bank0_trig_source = 1;

3. Choose trigger input signal type. Can be NIM or ECL. When a NIM trigger is available, set reg 0x6068 to 0 (NIM input is used for trigger).

4. Choose timing:
   Window start, for example you need 1 us for experiment trigger generation, so your trigger is 1 us delayed. The window start should be at –1100 ns in the past and should last for 256 ns. So set 0x6050 bank0_win_start = 16384 – 1100 = 15284 (ns)
   Set width 0x6054 bank0_win_width = 256 (ns).
   For **Resolution**, you may choose a channel width of 62.5 ps (set 0x6042 = 6), so the data will fill a 4 k spectrum. If you need higher resolution, a 3.9 ps channel width (set 0x6042 = 2) will fill a 64 k spectrum.

5. See chapter **"The MTDC32 read out"** to initialise the readout section of the MTDC

Special modes

The MTDC-32 supports also a full time stamping mode. In this mode every signal at the 34 inputs immediately creates an event for its own, which provides a 46 bit time stamp, based on the 3.9 ps resolution (1000/256 ps). It runs 275 s before it restarts
at 0. The full time stamp can be reset to 0 by register 0x6090 (reset_ctr_ab, external reset, single shot reset). This mode is especially for timing applications, which exceeds the 16 us window limit of the standard mode. The amount of data which is generated, may be up to a factor of 4 higher than in standard mode. Each edge on any input generates 4 x 32 bit words. The usual setting is reg 0x6038, marking_type is set to 3 (extended time stamp). Then the data format is identical to the standard format, but the end of event word contains the high part of the 46 bit timing. In standard mode it would be the event synchronisation counter low part. The extended time stamp instead contains the 16 event synchronisation counter lower bits.
When "marking_type" is set to 2, the ext. time stamp contains the lower 16 bits of the event counter, the other data do not change.

1 x header

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				# of following words											
01		0						0								15				3											

1 x data

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								T, channel #						TDC data (16 valid bits)															
0	0	0	0	0	1	0	0	0	0	t, chan-addr.						lower 16 bits of full time stamp															

1x "ext. time stamp" event time stamp, shows 16 low bits of the event synchronisation counter

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														Event sync. stamp															
0	0	0	0	0	1	0	0	1	0	0	0000					Event sync. stamp															

1 x end of event

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																															
1	1	higher 30 bits of full time stamp																													

T = trigger flag. Trigger flag is the address extension to "channel_number" to address the two trigger inputs. 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.
So the full time stamp is reconstructed by concatenating the 30 high bits of "end of event" and 16 low bits of "TDC data" to a 46 bit number.
In full time stamping mode, the window parameters, joined bank and some others have no effect. The single event mode (0x6036, multi event = 0) does not make much sense. In this mode, at high rates, there may arrive several events at the main FIFO before it is blocked for readout.

The MTDC-32 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)

Initialise 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 the conversion time of the TDC.

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 synchronisation

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 oscillator clock (for MTDC-32 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 to a reset register, or by an external reset signal.
All incoming events are then labelled 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 analyse and correlated events are grouped for further processing.

The synchronisation 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 MTDC-32 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 MTDC-32 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 MTDC 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 oscillator and will be reset (synchronised) via VME command.

Set oscillator input	ECL_gate1_osc	0x6064 = 1;
Set oscillator source, reset source	ts_sources	0x6096 = 2; (ext osc, int reset only)
Show time stamp in EOE mark	marking type	0x6038 = 1;
Synchronisation:	Reset_ctr_ab	0x6090 = 3; reset all counters

3. **IRQ**
   Initialise 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 initialisation, 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 MTDCs 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:

TDC1 in slot 1 gets 0x0100
TDC2 in slot 2 gets 0x0200
TDC3 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 initialise the individual modules:

TDC1: set 0x0100 6020 to 0xA2 (CBLT first module, Multicast enable)
TDC2: set 0x0200 6020 to 0x82 (CBLT mid module, Multicast enable) also any further module in the middle of the readout chain is initialised this way.
TDC3: 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 initialisation 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.

Using the two banks independently

The MTDC 32 can work as two independent 16 channel TDCs. In this mode it creates independent event structures for the two banks while the 5 bit (0..31) channel numbers are kept in the data words.

MTDC-32 overview

Address coders,
programming connector
jumper positions