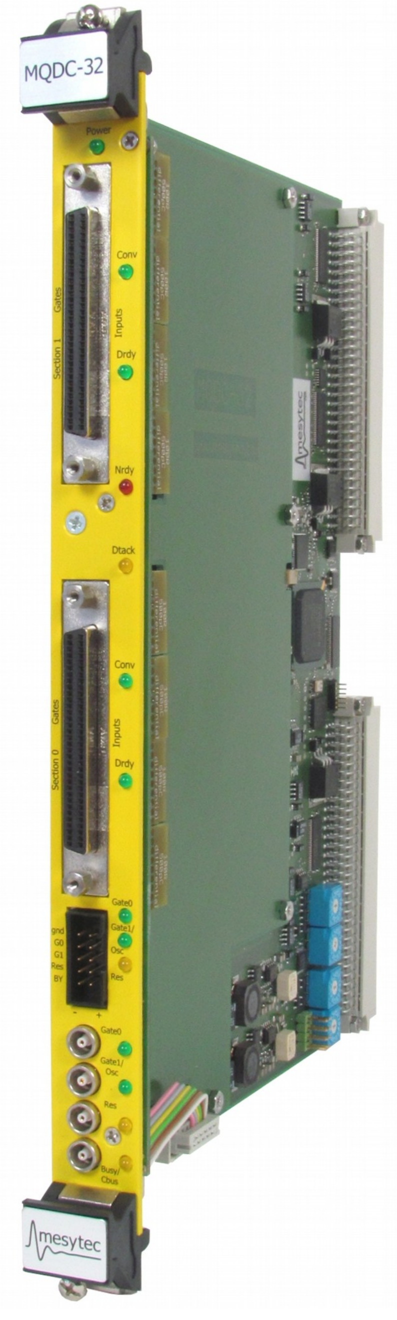
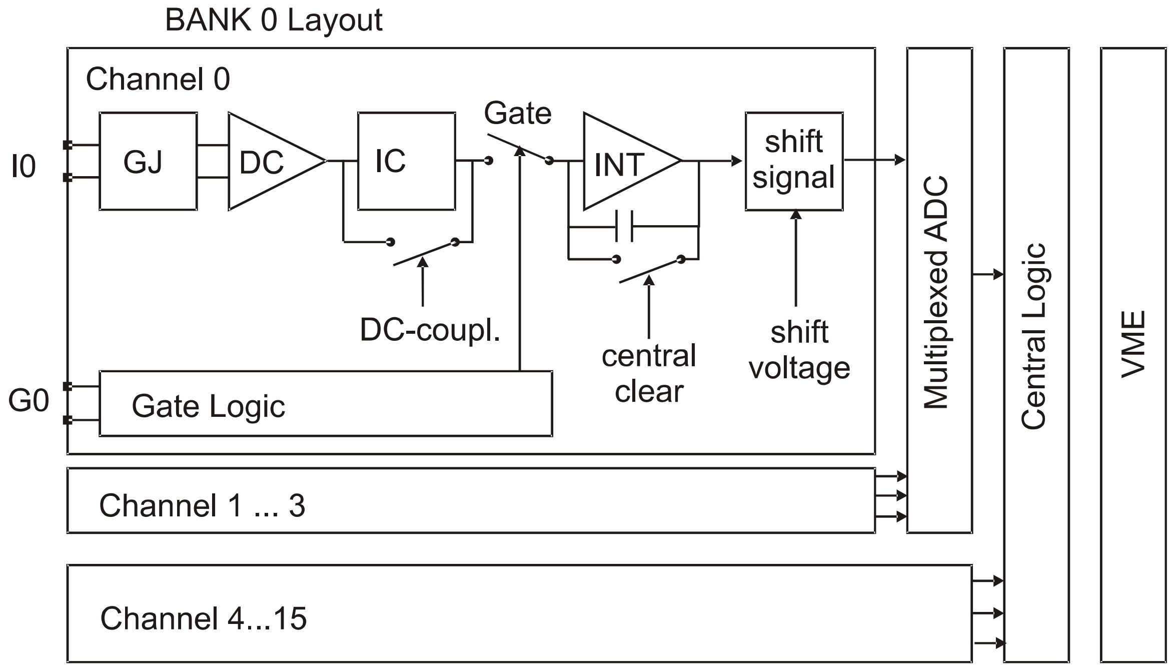
mesytec **MQDC-32** is a fast 32 channels charge integrating ADC with individual gates. It provides a 12 Bit (4 k) resolution with low differential non linearity due to sliding scale method. The total dead time is 250 ns for an event with all 32 channels responding. It supports zero suppression with individual thresholds. Special features: Built in pulse shape discrimination with 32 gate generators. Delayed gating with experiment trigger possible.

Features:

- High quality 12 bit (4 k) conversion with sliding scale ADC (DNL <2 %)
- 250 ns conversion and clear time for 32 channels
- Delayed gating with experiment trigger
- 64 k (32- bit-) words multi event buffer
  (1 word corresponds to 1 converted channel
  → 2 k to 21 k events total)
- Multiplicity filter, selects events on specified mult.-range
- Configurable: Individual gates or common gate
- Zero suppression with individual thresholds
- Supports different types of time stamping
- Independent bank operation
- Polarity is jumper selectable
  (for processing dynode or anode signals)
- Sensitivity and input resistance can be configured by special gain jumpers (50 pC to some nC. Default 500 pC)
- Signal input unipolar or differential (jumper configurable)
  Differential inputs allow easy delay by twisted pair cables
- AC coupled and baseline restored inputs.
  Optional DC-coupled via register setting
- Easy to use pulse shape analysis capability by 32 individual gate generators (4.5 ns to 300 ns)
- Gate and differential control inputs ECL, LVDS and PECL
  Can be terminated via register setting
- 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)
- low power consumption (12 W)



MQDC-32, Internal design



Input (I0) requires an analogue signal, for example from a photo multiplier. The signal may be differential or unipolar, the polarity may be positive or negative.

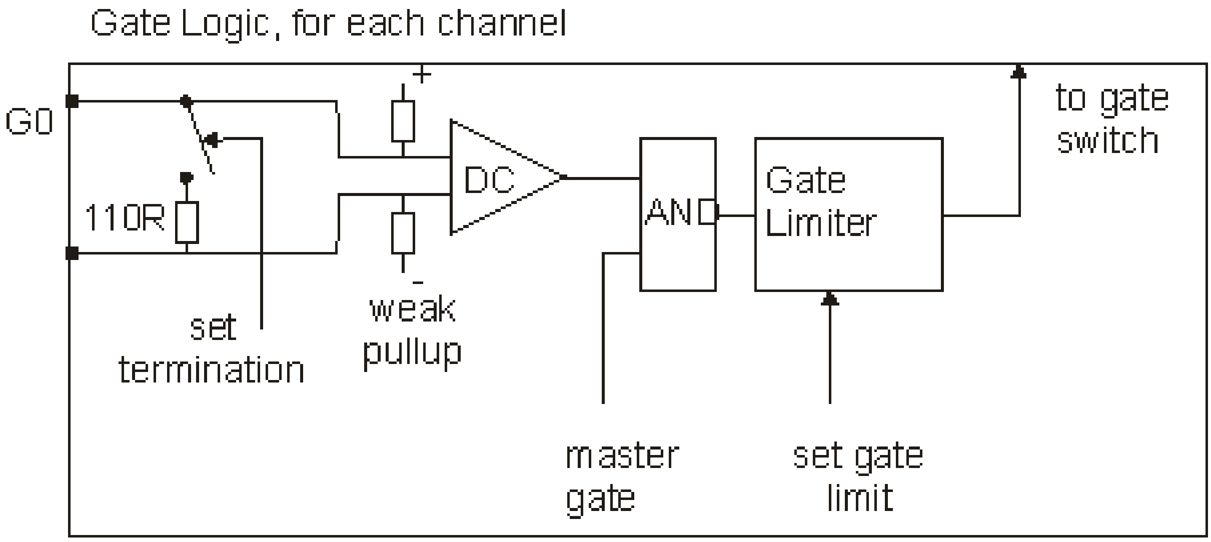
GJ is a small plug on module (gain jumper). It can be plugged on in two positions, coding for positive or negative input polarity.
There are different gain jumpers for differential or unipolar input, different input sensitivities and also input termination is coded on the jumpers.

DC is a fast linear amplifier and buffer unit.

IC is an AC-coupling, which filters out possible DC-offset of the input signal. To keep baseline stability at high rates, it also performs a baseline restoration based on the master gate. It can be bypassed by register setting.

Gate is a fast switch, which can open and close within 2 ns depending on the logic state of the gate line. Gates as short as 4 ns are possible.

INT is an integrator, which integrates the input signal when gate is on. It allows to integrate positive and negative signals. So undershoots of the input signals do not create problems. It is cleared after master gate and conversion are finished.
shift signal: it adds a voltage to the integrator output, before it is digitized. So the integrator output can be shifted to allow also conversion of integrals of slightly negative pulses.
Multiplexed ADC four channels are digitized by one ADC channel (25 ns conv. time).
All 32 channels are converted in 100 ns by 8 ADCs.
Central Logic is an FPGA with 64 000 words of memory. It controls the conversion and serves the VME-bus.
Gate Logic, each channel includes a gate logic, which controls the gates. See next diagram



Gate inputs (G0) are differential inputs, and work with any input in the range of ±4 V.
For example ECL and LVDS.

Set termination is an electronic switch, which can be set for each bank by register.
It allows to terminate the inputs with 110 Ω for the differential lines (default).

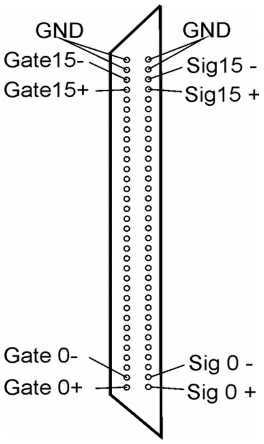
Weak pull up are two resistors which keep the gate input in a defined state when not connected.
Default is "on" state to allow the use of master gate without individual gates connected.

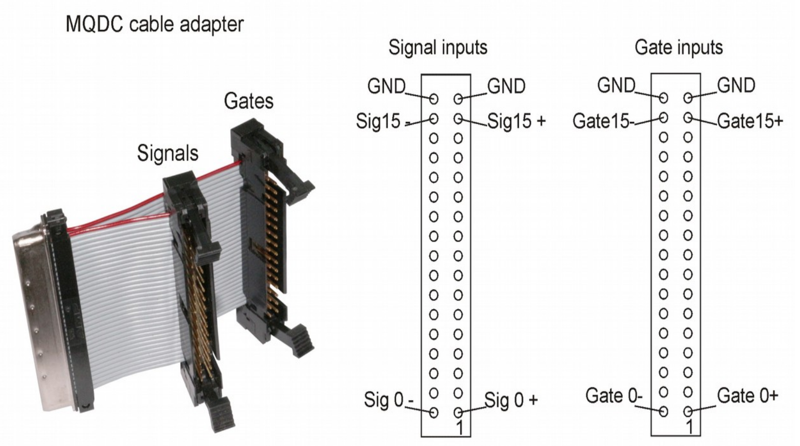
Gate Limiter is a circuit which can close the gate after an adjusted time. The time can be set in the range of 4 ns to 300 ns (or infinite) for each bank. It allows for easy pulse shape discrimination
(see section pulse shape discrimination).

Analogue channel Inputs

MQDC Input connector

Tyco AMPLIMITE. 050 Series, mates with connector: Tyco part_no 5749621-7
Two adapters from module connector to two 34 pin header connectors are included at delivery. Shown is the pin out of the lower (bank 0) connector. The upper one starts with channel 16 up to channel 31 (bank 1).

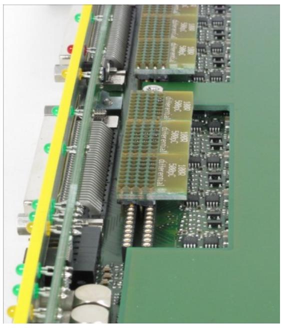




When unipolar gain jumpers are used, the negative signal inputs are internally connected to ground. With unipolar input, a mesytec adapter from 34 pin header connector to Lemo (Adapter MAD-34-16-S) can be added.

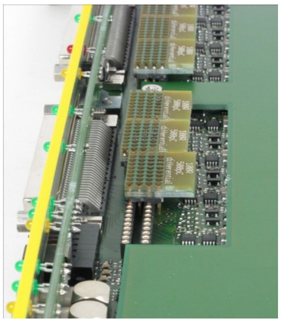
Analogue signal input configuration

Analogue signal inputs are configure by gain jumpers, which allow a high flexibility.
The jumper position codes for positive or negative inputs. The lowest jumper in the following pictures illustrate the polarity setting:



inverted input

(neg. signals at Sig+ input)



non inverted input

(pos. signals at Sig+ input)

Different types of gain jumpers are available, which code for

- unipolar or differential input
- input termination (high ohmic, 50 Ω, 110 Ω or custom type)
- input sensitivity (50 pC to some nC full range)
- standard jumpers
  termination: unipolar 50 Ω, differential 110 Ω, or high ohmic for pulse shape discrimination
  sensitivity: 50 pC, 150 pC, 500 pC, 1.5 nC, 5 nC

At delivery two sets of gain jumpers are included

Differential, 110 Ω, 500 pC
Unipolar, 50 Ω, 500 pC

Individual Gates input

The individual gate inputs accept differential signals of almost every differential standard. So ECL, PECL and LVDS is compatible. The inputs can be set as terminated 110 Ω or high ohmic by VME- registers.

Control input / output

Differential control inputs

interface any differential signals: ECL, LVDS or PECL.
They can be individually terminated via register setting

input/output	direction	termination	Default functionallity	Alternate functionalities
ECL3	Input	RT*	Gate0	-
ECL2	Input	RT*	Gate1	Time stamp oscillator input
ECL1	Input	RT*	Reset time stamp counter	Experiment Trigger Input
ECL0	Output	100 R	Busy	-
NIM3	Input	50 R	Gate0	-
NIM2	Input	50 R	Gate1	Time stamp oscillator input
NIM1	Input	50 R	Reset time stamp counter	Experiment Trigger Input
NIM0	Input / Output	50 R	Busy	Mesytec control bus Buffer full Data conv ready FIFO data threshold reached

** "RT" means register selectable termination.*

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)

Front Panel LEDs

LED „Conv“	digitization in progress
LED „Drdy“	Data are ready converted and can be read out
LED „Nrdy“	Gate detected, but ADC busy. Event will be lost
LED „Dtack“	Access from VME bus accepted

Lemo and differential inputs

minimum gate width for master- and for individual gates = 4 ns

maximum external reference oscillator frequency: 75 MHz

Signal processing and conversion

The basic difficulty for QDC operation is baseline offset and baseline drift. MQDC-32 overcomes this by providing AC coupled inputs with baseline restoration. The signals are restored while the master gate is not active. The restoration works with a long time constant to avoid any influence on the fast signals.

For special applications DC coupling is possible by register setting.

The internal amplifiers of MQDC-32 preserve a signal rise time of 2 ns. The high band width allows precise pulse shape analysis of very fast signals. The internal charge integrator accepts positive and negative signals, so also signals with negative undershoot (for example due to reflections) can be processed. The integrator output can be shifted by ±25 % of the full range before it is digitized. Sliding scale method is used to achieve a low differential non linearity. It needs 1/16 of the full 4 k binary value.

Measurement conditions: gate of 50 ns

Conversion	Typical Noise in channels rms	Max noise Channel rms	Conversion time 32 channels	highest channel / overflow channel
4 k	1.2	1.6	100 ns total dead time: 250 ns	3839 / 3840

After conversion, additional filters are available

- Threshold for individual channels:
  if the thresholds are set, signals below the thresholds are suppressed in the output data.
- Lower and upper threshold for the total number of responding channels (multiplicity) are available. It the number of responding channels is not within this window, the events are suppressed.

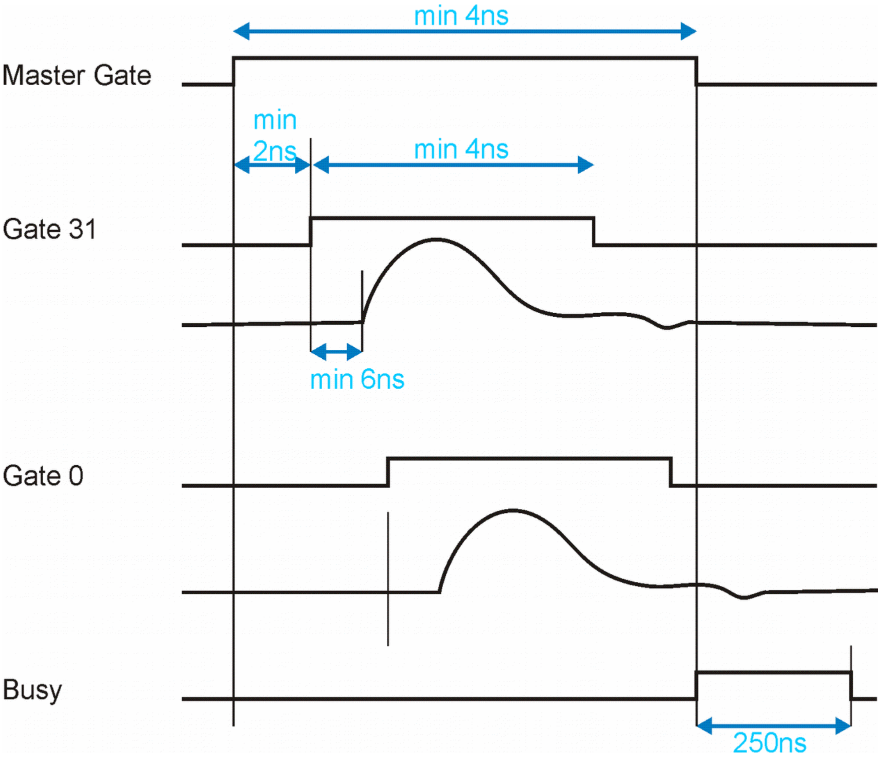
Conversion, busy time

- Digital conversion time: 100 ns
- Conversion starts 50 ns after gates close
- Recovery time after conversion: 100 ns

Gate operation

The MQDC-32 allows to work with individual gates and also with only one or two master gates (Two gates at independent bank operation).
This mode is established when the individual gate inputs are left floating. By default they are internally pulled to the on state.

Gate and analogue signal timing



The individual gates should precede the signal by minimum 6 ns.
Master gates should preceded the individual gates by at least 2 ns.
Minimum gate length for master gates and individual gates is 4 ns.

Delayed gating with experiment trigger

As MQDC-32 is a very fast converter, it can be usually operated free running with the common "OR" output of a CFD (works for example at 400 kHz free rate per gate with 10 % dead time). When using MCFD-16 ored trigger as master gate, only short twisted pair delay cables are needed to fulfil timing requirements.
But usually the ored free running triggers of a CFD can not be used to trigger all digitization modules of a setup. This would produce excessive data and dead time. A central trigger logic is needed to decide from all detector triggers if the event data should be converted or not. This trigger is called here "**experiment trigger**". Such a trigger decision can only be taken when the slowest detector of the setup, which is relevant for the trigger decision, has produced its trigger. This delay may be for example a microsecond.
Now the typical problem with QDCs emerges.
The decision if a QDC-gate has to be created depends on experiment trigger, so all signals to the QDC have to be delayed by a microsecond.
This will result in a cable length of 200 m or expensive delay modules, which degrade the analogue signals.
In delayed gating mode, MQDC-32 overcomes this by sending its converted data in a queue.
A delay can be defined by register, in this example delay 1100 ns. If the experiment trigger signal is high at that delay time relative to the event stamp (created at conversion gate end), the event may pass to the main buffer, else it is skipped.
This is a big advantage over the traditional fast clear. It does not create additional dead time, as many events can be stored in the queue (FIFO), and events can be picked selectively in a very short coincidence time (experiment trigger width, minimum 25 ns wide).
The coincidence time widow can be determined by generating the correct experiment trigger width at the QDC input.

The delays can be set at the following registers

reg 0x6054 daq_gate_del0 in ns, off when = 0
reg 0x6056 daq_gate_del1 in ns, off when = 0
(at split banks only)

The reset inputs can be selected as experiment trigger inputs by setting the registers

Experiment trigger available as ECL signal
reg 0x6066 ecl_fc_res = 2 ECL reset input used

Experiment trigger available as NIM signal
reg 0x606C nim_fc-reset = 2 NIM reset input used

As there is only one experiment trigger, there are no different gate inputs for the banks at split bank operation.

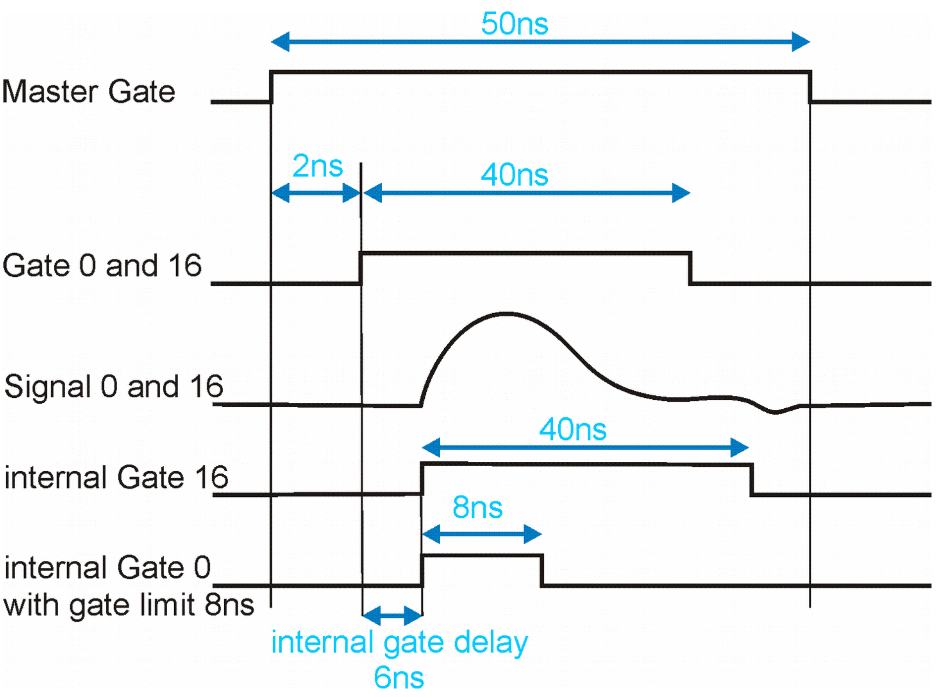
Pulse shape discrimination

MQDC-32 provides several features to allow easy use as a very fast pulse shape discriminator.
Pulse shape discrimination is for example needed for particle discrimination in liquid scintillators. In this case an integral over the peak of the pulse is compared to the integral of all the pulse.
For particles, which create a low charge density in the scintillator, a component with short decay time is dominantly emitted, for particles with high ionisation density only a slower component is emitted.

The total setup consists of a fast constant fraction discriminator with adjustable trigger output length and fast ored output of the individual triggers.
The signals are delayed adequately and fed to the MQDC-32.

The analogue and gate signals are connected to the 2 banks of MQDC-32 in parallel. The gate termination for one bank is switched off, the gain jumpers of one bank are replaced with high ohmic ones.

The MQDC-32 provides **Gate limiters** for each channel which allow via register setting to limit the input gates to a value between 4 ns and 300 ns.
The gate limit is now used for bank 0
(limit set to 8 ns → reg 0x6050 = 19).
The picture below shows how the internal gates are modified.
The integral of channel 0 (bank 0) can now be divided by the integral of channel 16 (bank 1) and thus gives a measure for the ionisation density, so a particle discrimination value.



Use MQDC-32 with MCFD-16

Pulse shape discrimination and all QDC operation gets very easy when using the MCFD-16 and MQDC-16 as a pair. The necessary delays are then quite short, and can be established with a few meters of twisted pair cables. Differential gain jumpers are used, and set to non inverting position. There is in many cases no need for further external modules.

The cable length can be calculated the following way

- L0 = Lemo cable length from fast OR of MCFD-16 to Gate0 input of MQDC-32
- L1 = 34 pin Twisted pair cable from ECL pulse output MCFD to Gate inputs MQDC
- L2 = 34 pin twisted pair cable from MCFD analogue output to MQDC signal inputs

Example

Signals from photo multiplier with 8 ns (20 % to peak) rise time are fed to the MCFD-16

The fraction is selected as 0.2 (20 %)

The tap delay is set to tap 2, with 20 ns delay chips this results in 20/5*2 = 8 ns

For L0 = **1 m** of cable is needed

Cable length

- L1 = L0 + 2 m = **3m**
- L2 = L1 + 2 m + (tap_delay [ns] * (1 + fraction) / 5 [m/ns]) = **7 m**

Power consumption

(Total: 12 W)
+ 5 V +1.0 A
+12 V +300 mA
–12 V –250 mA

MQDC32 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).
memory size: 64 k-72 = 65464 words with 32 bit length

Header (4 byte)

2 header signature	6 subheader	8 module id	1 no meaning	3 fill bits	12 number of following data words, including EOE
b01	b000000	module id	bx	b000	number of 32 bit data words

Data (4 byte) DATA event

2 data-sig	9	5	1 out of range	3	12
b00	00 0100 000	Channel number	Oor	b000	ADC amplitude

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	1	3	12
b00	0	0	0	0	0

End of Event mark (4 byte)

2	30
b11	trigger counter / time stamp

Threshold memory at address x4000 to x403F (16 bit words, access: R/W D16)

Address	Name	Bits	dir	Default	Comment
0x4000	threshold[0]	12	RW	0	Threshold value of channel 0 value 0 = threshold not used
...					
0x403E	threshold[31]	12	RW	0	Threshold value channel 31

With the value 0x1FFF the channels are switched off.

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 (board coder).
0x6006	fast_vme	1	RW	0	accelerated VME interface, set to 0xfa up to 20 % faster access*
0x6008	soft_reset	1	W		breaks all activities, sets critical parameters to default. Wait 1 s until all settings are stable.
0x600E	firmware_revision	16	R		Format: 0xYY.ZZ, Y and Z: 0...F

	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.

** new in firmware FW0200*

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

0x6020: CBLT_MCST_Control 1

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 + 16 bit module address space

MCST Address Field

A31.............A24	**A23..A00**
MCST ADRS	8 high bits, not significant + 16 bit 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.
Usually when zero suppression is used and all modules were gated, each Module emits the header and footer with time stamp (2 Words with 32 bits each: MQDC-32 Header, MQDC-32 footer).

<table>
<tr><th></th><th>FIFO handling</th><th></th><th></th><th></th><th></th></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
The number of 32bit 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.
If set to = 4 events in the FIFO are shown *</td></tr>
<tr><td>0x6034</td><td>readout_reset</td><td></td><td>W</td><td></td><td>At single event mode (multi event <b>= 0</b>): allow new trigger, allow IRQ
At multi event <b>= 1</b>: checks threshold, sets IRQ when enough data. Allows safe operation when buffer fill level does not go below the data threshold at readout.
At multi event <b>= 3:</b> 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><b>mode</b>[1:0]</td></tr></table>
allow multi event buffering (bit 0,1)
<b>mode = 0</b> → <b>no</b> (0x6034 clears event, allows new conversion)
<b>mode = 1</b> → <b>yes</b>, unlimited transfer, no readout reset required (0x6034 can be written after block readout).
Don’t use for CBLT
<b>mode =3</b> → <b>yes</b> but the module transfers limited amount of data. With reg 0x601A the number of data words can be specified. After word limit 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).

<b>Berr handling:</b> 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>
</table>

0x6038	marking_type	2	RW	0	00 → event counter (starts with 0 at first event) 01 → time stamp 11 → extended time stamp
0x603A	start_acq	1	RW	1	1 → start accepting gates 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 buffer is in a well defined status.
0x603C	fifo_reset		W		initialise FIFO
0x603E	data_ready	1	R		1 → data available

** new in firmware FW0200*

	operation mode				
0x6040	bank_operation	3	RW	b000	bit 0: 0 → banks connected 1 → operate banks independent bit 2, 1: bank 1, 0 switch weak pull on resistor off (usually leave bits at 0, 0)
0x6042	adc_resolution	1	RW	fixed 0	0 → always 4 k
0x6044	Offset_bank_0	8	RW	Per*	Channels 0 to 15: Range 0 to 255, Shifts 4 k spectrum by ±1000 bins
0x6046	Offset_bank_1	8	RW	Per*	same for channels 16 to 31
0x6048	slc_off	1	RW	0	Switch off sliding scale
0x604A	skip_oorange	1	RW	0	Skip out of range values
0x604C	Ignore Thresholds	1	RW	0	Set at 1 threshold settings at 0x4000... are ignored (assumed as 0)

* Modification of these registers may take some milliseconds to get effect in the hardware.

	gate limit				
0x6050	limit_bank_0	8	RW	per*	4 = 4.5 ns, 255 = no limitation
0x6052	limit_bank_1	8	RW	per*	same as for bank 0
	experiment trigger				**experiment trigger is switched on when ECL1 input (reg 0x6066) or NIM1 input (0x606C) is set to = 2**
0x6054	exp_trig_delay0(1)	14	RW	0	delay of experiment gate relative to end of QDC-gate (ns). Up to 16384 ns (16 k) For bank0 or both at joined bank
0x6056	exp_trig_delay1(1) *(1) only available in firmware revision FW0110 and higher, all devices upgradable*	14	RW	0	For bank 1 if not joined

Limit Bank 0/1 lookup table

DAC	time [ns]
4	4,5
5	4,6
6	4,6
7	4,7
8	4,8
9	4,8
10	4,9
11	5,0
12	5,1
13	5,2
14	5,3
15	5,4
16	5,6
17	5,7
18	6,1
19	7,6
20	8,7
21	9,4
22	10,1
23	10,9
24	11,6
25	12,3

DAC	time [ns]
30	16
35	19
40	24
50	32
60	41
70	51
80	62
90	73
100	86
110	101
120	117
130	136
140	158
150	185
160	219
170	268
175	300

IO 0x6060	**Inputs, outputs**				
0x6060	input coupling	3	RW	b000	bit[0] bank0: 0 → AC, 1 → DC bit[1] bank1: 0 → AC, 1 → DC bit[2] 0 = BLR on, 1 = BLR off (FW0114)
0x6062	ECL_term	5	RW	b11000	switch ECL/LVDS terminators on (1= on) bit 0 for: "gate0", bit 1 for "gate1", bit 2 for "Res", bit 3 for bank 0 individual gate input termination, bit 4 for bank 1 individual gate input termination. Unconnected inputs will be in a well defined state by internal weak pull up resistors.
0x6064	ECL_gate1_osc (ECL2)	1	RW	0	0 → gate1 input, 1 → oscillator input (**also set 0x6096!!**)
0x6066	ECL_FC_Reset (ECL1)	2	RW	1	0 = Fast clear input (= "not Gate") 1 = Reset time stamp counter 2 = input for experiment trigger (1)
0x6068	Gate_select	1	RW	0	0 → Gate 0 and 1 from NIM-inputs, 1 → Gate 0 and 1 from ECL-inputs
0x606A	NIM_gat1_osc (NIM2)	2	RW	0	0 → gate1 input, 1 → oscillator input (**also set 0x6096!!**)
0x606C	NIM_FC_Reset (NIM1)	2	RW	0	0 = Fast clear input (= "not Gate") 1 = Reset time stamp counter 2 = input for experiment trigger ***(1)***
0x606E	NIM_busy (NIM0)	4	RW	0	0→ as busy (in independent bank operation: active when both banks are busy) 3→ as Cbus output 4→ buffer full 8→ data in buffer above threshold 0x6018 9→ events in buffer above threshold 0x601E(2)

(1) only available in firmware revision FW0110 and higher, all devices upgradable
Experiment trigger gets active when ECL or NIM are configured as exp trigger input.
Starting with FW0200 ECL3 is also event gate input when 0x606C = 2 and 0x6068 = 0 (ECL3 not configured as gate0 input). Simplifies cabeling of ECL bus withh MQDC and MTDC in the chain.
(2) new in firmware FW0200

0x6070	**Test pulser**				
0x6070	pulser_status;	4	RW	0	b000 = off, b100 = amplitude = 0, b101 = use pulser-amplitude with 100 ns gate
0x6072	pulser_dac	8	RW	32	Pulser amplitude, typ. 32 for bin 2000

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 Gate0-LED shows data traffic on the bus, the Gate1-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 MQDC32 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 MQDC32 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
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
For counters "ADC_busy" and "Gate1_busy" the count basis is 25 ns.
Output value is divided by 40 to give a 1 us time basis

CTRB 0x60A0	**counters B**				
0x60A0	adc_busy_time_lo	16	R		ADC busy time, from gate to end of conversion. Step [1 us]
0x60A2	adc_busy_time_hi	16	R		
0x60A4	gate1_time_lo	16	R		Gated time from Lemo gate 1 input [1 us] timer counts when gate1 has active NIM level (–0.6 V). Step [1 us]
0x60A6	gate1_time_hi	16	R		
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	6	RW	32	Bank0 (or B0, B1 when connected) upper limit of responding channels
0x60B2	low_limit0	6	RW	0	Bank0 (or B0, B1 when connected) lower limit of responding channels
0x60B4	high-limit1	5	RW	16	Bank 1 upper limit
0x60B6	low-limit1	5	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 64 k x 32 bit.

Data is organized in an event structure, maximum size of one event is 36x 32-bit words (Header, End of event, 32 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								x	fill			# of following words											
0	1	0	0	0	0	0	0	ii	ii	ii	ii	ii	ii	ii	ii	x	0	0	0	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
fill:	fill bits, b000	
x:	may be 1 or 0, no meaning	
# 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									channel #					V		ADC data (12 valid bits)													
0	0	0	0	0	1	0	0	0	0	0	c	c	c	c	c	v	0	0	0	d	d	d	d	d	d	d	d	d	d	d	d

dsig: data signature = b00

fix: bit field currently without meaning = b000100000 → Byte #3 = 0x04

channel #: number of ADC channel → Byte #2 = channel#
within an event buffer, ADC channels may occur in arbitrary order

V: V = 1 indicates ADC over- or underflow

ADC data: ADC conversion data, data width 12 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 (30 bit)																													
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, MQDC-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, MQDC-32 emits a VME Berr (bus error)

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

The MQDC32 read out in two modes

Single event readout

In this mode the pulses are stretched and converted starting with an external gate or 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 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, multivalent = 1 or 3) the input is decoupled from output by a 64 k words buffer. So the input is ready for a new gate after the conversion time of the ADC.
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 get asynchronous.

The better one is **time stamping.**
A central oscillator clock (for MQDC32 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 29 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 MQDC32 allows to set a buffer fill threshold which emits an interrupt when the data fill level in the buffer exceed 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, because at fastest conversion the dead time may be as low as 200 ns, the amount of data without zero suppression may be 34 words per conversion. So the theoretical amount of data written to the buffer can be up to 170 Mwords/s, the VME readout rate is realistic about 5 Mwords /s in BLT32. So under worst conditions it is not possible to empty the buffer via VME and get an empty FIFO signal “Berr”! So if high rates can appear, the data acquisition should at least be tolerant to splitted events.
3. 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 MQDC32 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 MQDC 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 MQDC modules in 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) Set Multi event

Multi event 0x6036 = 3	multi event with limited data transfer
Irq_threshold 0x6018 = 200	Irq is set when more than 200 (32 bit-)words are in buffer
Max_transfer_dat 0x601A = 222	transmit maximum 222 words + rest of event before sending Berr. (In this case data fits into one VME 255 word blt32transfer)

4) 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 100 (IRQ emitted when more than 100 words in FIFO)

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 MQDCs 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:
ADC1 in slot 1 gets 0x0100
ADC2 in slot 2 gets 0x0200
ADC3 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:

ADC1: set 0x0100 6020 to 0xA2 (CBLT first module, Multicast enable)
ADC2: 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.
ADC3: 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 to preserve even word numbers.

CMBLT64

Is intrinsic when chained block transfer is used with MBLT64.

Using the two banks independently

The MQDC-32 can work as two independent 16 channel ADCs. 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.

MQDC-32 overview

Address coders,
input gain jumpers