mesytec **MADC-32** is a fast and high quality 32 channels peak sensing ADC. It provides an 11 to 13 Bit (2 to 8 k) resolution with low differential non linearity due to sliding scale method. In 8 k mode it provides the high INL and resolution required for Ge detector readout. The conversion time is 800 ns for 32 channels at 2 k resolution. It supports zero suppression with individual thresholds.

Features:

- High quality 11 to 13 bit (2, 4, 8 k) conversion with sliding scale ADC (DNL <1 % @ 4k)
- 800 ns, 1.6 us, 6.4 us conversion time for 32 channels wit 2 k, 4 k, 8 k resolution
- 8 k (32 bit-)words multi event buffer (1 word corresponds to 1 converted channel => 240...2730 events total)
- Zero suppression with individual thresholds
- Supports different types of time stamping
- Independent bank operation
- Two register adjustable gate generators are built in
- Input range, register selectable 4 V, 8 V, 10 V
- 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)

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

Technical Data

Input / Output

Conversion input 1 kΩ, 4 V, 8 V or 10 V configurable via register
Rise time min.: 50 ns, max: DC-conversion possible

ECL inputs:
standard ECL input, can be individually terminated via register setting

NIM inputs:
standard NIM

NIM output:
–0.7 V terminated

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*	Gate 0	-
ECL2	Input	R	Gate 1	Time stamp oscillator input
ECL1	Input	R	Fast clear	Reset time stamp counter
ECL0	Output	100 R	Busy	-
NIM3	Input	50 R	Gate 0	-
NIM2	Input	50 R	Gate 1	Time stamp oscillator input
NIM1	Input	50 R	Fast clear	Reset time stamp counter
NIM0	Input / Output	50 R	Busy	2) mesytec control bus I/O for monitoring and adjustment: 3) internal gate generator 0 output 4) internal gate generator 1 output

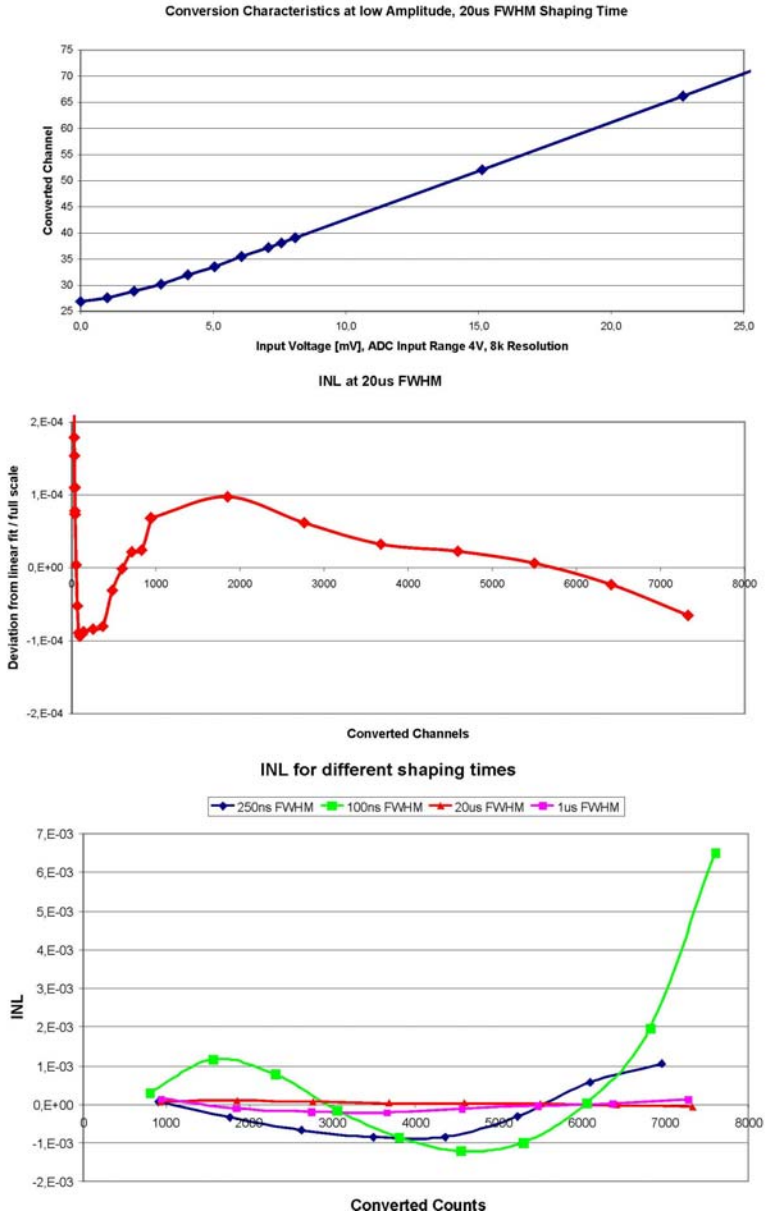
"R" means register selectable termination.

Front Panel LEDs

LED "Conv"	digitization in progress
LED "Drdy"	Data are ready converted and can be read out
LED "Nrdy"	Gate detected, but ADC busy. Event will be lost (should never happen in a well controlled DAQ). Or fast clear outside of acceptance window.
LED "Dtack"	Access from VME bus accepted

Integral non linearity

INL @ 20 us FWHM shaping time typ. 1*10-4 (max 2*10-4) in the range 0.2 % to 100 %
INL @ 1 us FWHM shaping time typ. 2*10-4 in the range 0.2 % to 100 %
INL @ 0.25 us FWHM shaping time typ. 1*10-3 in the range 1 % to 100 %



Output formats, resolution, conversion time

The sliding scale needs 1/16 of the full binary value.

Conversion	Typical noise in channels rms	Max noise in channel rms	Conversion time 32 channels	highest channel / overflow channel
2 k	0.6	0.8	0.8 us	1919 / 1920
4 k	0.9	1.2	1.6 us	3839 / 3840
4 k hires	0.7	1.0	3.2 us	3839 / 3840
8 k	0.9	1.2	6.4 us	7679 / 7680
8 k hires	0.7	1.0	12.8 us	7679 / 7680

Power consumption (Total: 4.0 W)

+5 V, + 190 mA
+12 V, + 160 mA
–12 V, – 80 mA

Conversion, busy time

Digital conversion time:
800 ns for 32 channels @ 2 k resolution
Conversion starts 50 ns after gates closes
Recovery time after conversion: 200 ns

Lemo and ECL inputs

Minimum gate width: 80 ns
Minimum clear signals: 50 ns
Maximum external reference oscillator frequency: 75 MHz

Gate generators

Two gate generators are provided for each bank of 16 channels.
When the gate generators are used, the gate inputs work as trigger inputs to start the gate generators. When active the gate generators create a gate for the stretchers. Delays and widths can be adjusted independently in steps of 50 ns. Minimum delay (gate_delay parameter = 0) is 25 ns.
For monitoring, the gate signals can be switched to the busy output via register setting
(reg. 0x606E).

MADC32 register set

Data FIFO, read data at address 0x0000 (access R/W D32, 64)
for 64 bit access, last 32 bits will be filled with 0 for odd number of 32 bit data words
memory size: 1024 + 2 = 1026 words with 32 bit length

Header (4 byte)

2 header signature	6 subheader	8 module id	1 output_ format → 0x6044	3 adc_ resolution → 0x6042	12 number of following data words, including EOE
b01	b000000	module id	bx	bxxx	number of 32 bit data words

Data (4 byte) DATA event

2 data-sig	9	5	1	1 out of range	1..3	11..13
b00	00 0100 000	channel number	b0	Oor	b00	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	1	2	12
b00	0	0	0	0	0	0

End of Event mark (4 byte)

2	30
b11	event 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]	13	RW	0	Threshold value of channel 0 value 0 = threshold not used
...					
0x403E	threshold[31]	13	RW	0	Threshold value channel 31

Threshold values are referenced to the actually adjusted data output width (2 k, 4 k or 8 k).
Reg 0x6044, 0x6046. With the value 0x1FFF the channels are switched off in any resolution mode.

Registers, Starting at address x6000 (access D16)

Blue marking is implemented starting with firmware FW0220

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).
0x6008	soft_reset	1	W		breaks all activities, sets critical parameters to default
0x600E	firmware_revision	16	R		0x01.10

	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_threshold	13	RW	1	Every time the number of 32 bit words in the FIFO exceeds this threshold, an IRQ is emitted. Maximum allowed threshold is 8120.
0x601A	Max_transfer_data	14	RW	1	Maximum data words to transfer before ending the transfer at next end of event word. Only works for multi event mode 3. At Max_transfer_data = 1, 1 event per transfer is emitted. Maximum number of events is 2047. Usually the same or higher value than in 0x6018 is used. 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

0x6020: CBLT_MCST_Control

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: MADC32 Header, MADC32 footer).

	FIFO handling				
0x6030	buffer_data_length	16	R		amount of data in FIFO (only fully converted events). Units → data_len_format. Can be used for single- and multi event transfer
0x6032	data_len_format	2	RW	2	0= 8 bit, 1=16 bit, 2=32 bit, 3=64 bit At 3 a fill word may be added to the buffer to get even number of 32 bit words.
0x6034	readout_reset		W		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 multi event = **3** : clears Berr, allows next readout

<table>
<tr><td>0x6036</td><td>multi event</td><td>4</td><td>RW</td><td>0</td><td>
<table>
<tr><td>Bit[3]</td><td>Bit [2]</td><td>Bit[1:0]</td></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 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).

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
(now also for independent bank operation)</td></tr>
</table>

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 be-fore applying the FIFO_reset to get a well de-fined 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

	operation mode				
0x6040	bank_operation	2	RW	0	b00 → banks connected b01 → operate banks independent b11 → toggle mode for zero dead time (use with internal gate generators enabled)
0x6042	adc_resolution	3	RW	2	0 → 2 k (800 ns conversion time) 1 → 4 k (1.6 us conversion time) 2 → 4 k hires (3.2 us conversion time) 3 → 8 k (6.4 us conversion time) 4 → 8 k hires (12.5 us conv. Time)
0x6044	output_format	1	RW	0	0 → addressed mesytec format.
0x6046	adc_override	2	RW	2	When written, it overrides the channel output width (2 k...8 k) but not the conversion time. Values defined as in "adc_resolution"
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)

	gate generator				
0x6050	hold_delay0	8	RW	20	0= 25 ns, 1= 150 ns, then multiple of 50 ns
0x6052	hold_delay1	8	RW	20	same as for bank 0
0x6054	hold_width0	8	RW	50	multiple of 50 ns
0x6056	hold_width1	8	RW	50	Same as for bank 0
0x6058	use_gg	2	RW	0	01 = use GG0 10 = use GG1 (GG1 can only be activate when reg 0x6040 ! = 00, banks not connected)

IO 0x6060	**Inputs, outputs**				
0x6060	input_range	2	RW	0	input range: 0 → 4 V, 1 → 10 V, 2 → 8 V
0x6062	ECL_term	3	RW	b000	switch ECL terminators on (1= on) low bit for: “gate0”, high bit for "fc" Switch terminators off when inputs are not used. Then inputs will be set to a well defined state by internal weak resistors.
0x6064	ECL_gate1_osc (ECL 2)	1	RW	0	0 → gate1 input, 1 → oscillator input (**also set 0x6096 !!**)
0x6066	ECL_fc_res (ECL 1)	1	RW	0	0 → fast clear input, 1 → reset time stamp oscillator input
0x6068	ECL_busy (ECL 0)	1	RW	0	0 → as busy output, 1 → reserved

0x606A	NIM_gat1_osc (NIM 2)	1	RW	0	0 → gate1 input, 1 → oscillator input (**also set 0x6096 !!**)
0x606C	NIM_fc_reset (NIM 1)	1	RW	0	0 → fast clear input, 1 → reset time stamp oscillator, hold at value 0
0x606E	NIM_busy (NIM 0)	4	RW	0	0 → as busy (in independent bank operation or toggle mode: active when both banks are busy) 1 → as gate0 output 2 → as gate1 output 3 → as Cbus output *(needs up to 100 us. Read signals the correct status)* 4 → buffer full 8 → data in buffer above threshold 0x6018 9→ events in buffer above threshold 0x601E

IOs are counted from bottom to top (NIM0..3, ECL0...3)

0x6070	**Test pulser**				
0x6070	pulser_status;	4	RW	0	b000 = off, b100 = amplitude =0 (input-> 10 V range) b101 = low amplitude (7 %), b110 = high amplitude (75 %) b111 = amplitude: 0 -> low -> high → 0...

Mesytec control bus

Set 0x606E to 3 before using Cbus. Wait 100 us to allow updating of output configuration or poll register 606E for bit 1,0 update
As long as Cbus is activated, the NIM inputs Gate 0 and Gate 1 – osc get inactive

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		bit 0 = active bit 1 = address collision bit 2 = 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 bit 0 = 0 is possible

The Gate 0 – LED shows data traffic on the bus, the Gate 1 – LED shows bus errors
(i.e. non terminated lines)

Example for controlling external modules with mesytec RC-bus

Initialise and read out a MSCF-16 Shaper module.
MSCF-16 ID-coder set to 7
Bus line must be terminated at the far end.

Activate MADC-32 control bus at busy line

Write(16) addr 0x606E data 3

Get Module ID-Code (=Type of module = 20 for MSCF-16)

```
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
```

Set gain for channel 1 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 1     // address module memory location 1
Write(16)    addr 0x6088 data 10    // start send. Data to send
```

Wait loop: Read(16) 0x608A and compare bit 0 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 gain of channel 1

```
Write(16)    addr 0x6082 data 7     // address module 7
Write(16)    addr 0x6084 data 18    // code "read_data"
Write(16)    addr 0x6086 data 1     // 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 bit 0 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 MADC-32 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. Fast cleared events are not counted.

CTRA 0x6090	**countersA**				
0x6090	Reset_ctr_ab	4	RW	0	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. Read this bit gives information that counter was reset.
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 1µs time basis

CTRB 0x60A0	**countersB**				
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)

Data handling

The event buffer is organised as a FIFO with a depth of 8k x 32 bit.
Data is organized in an event structure, maximum size of one event is 36 32-bit words.

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								f	adc res			# of following words											
0	1	0	0	0	0	0	0	ii	ii	ii	ii	ii	ii	ii	ii	f	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

output format f: currently = 0 (addressed format)

adc res: ADC resolution, depending on register 0x6044
0 = 2 k, 800 ns conv. Time
1 = 4 k, 1.6 us conv. Time
2 = 4 k, 1.6 us conv. Time
3 = 8 k, 1.6 us conv. Time
4 = 8 k, hires, 3.2 us conv. Time

# 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 (11-13 valid bits)													
0	0	0	0	0	1	0	0	0	0	0	c	c	c	c	c	0	v	0	d	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 overflow

ADC data: ADC conversion data, data width 11...13 valid bits, depending on register 0x6044

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		event 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

event counter/ time stamp: 30 bit event 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, MADC-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, MADC-32 emits a VME Berr (bus error)
This can be used to terminate a block transfer or multi block transfer.

The MADC32 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, multi event = 1 or 3) the input is decoupled from output by an 8 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. Also fast clears may be necessary for some detectors. 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 MADC32 this can be the VME built in clock of 16 MHz or an exter-nal 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. 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 MADC32 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 1us, the amount of data without zero suppression may be 36 words per conversion. So the theoretical amount of data written to the buffer can be up to 36 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 MADC-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 MADC 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 MADC 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 (int reset only, ext osc)	ts_sources	0x6096 = 1;
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 blt32 transfer)

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 MADCs 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 Operation

MBLT64

MBLT64 is defined by the address modifier. At buffer unpacking in analysis software it helps to keep the alignment within the transmitted 64 bit word. When setting register 0x6032 to 3 a fill word is added to the converted channels when converted channel number is odd.

CMBLT64

Is intrinsic when chained block transfer is used with MBLT64.
Note: Register 0x6032 has to be set to 3 to align the words!

Using the built in gate generators

The MADC-32 provides built in gate generators (for joined bank operation only gate generator 0 is used). Only a short trigger has to be delivered to the gate input to start the gate generator. Delay and width of the gate can be set via registers.
To use the gate generators, they have to be defined as gate source in 0x6058. For monitoring of the generated gate, it is available at the busy output when setting 0x606E to 1 (gate 0) or 2 (gate1).

Using the two banks independently

The MADC-32 can work as two independent 16 channel ADCs. The gate 0 starts conversion of bank 0, gate 1 starts bank 1. In this mode the module creates independent event structures for the two banks while the 5 bit (0..31) channel numbers are kept in the data words.
All converted events from both banks are stored as they come in the main FIFO, and can be read out via VME-bus.

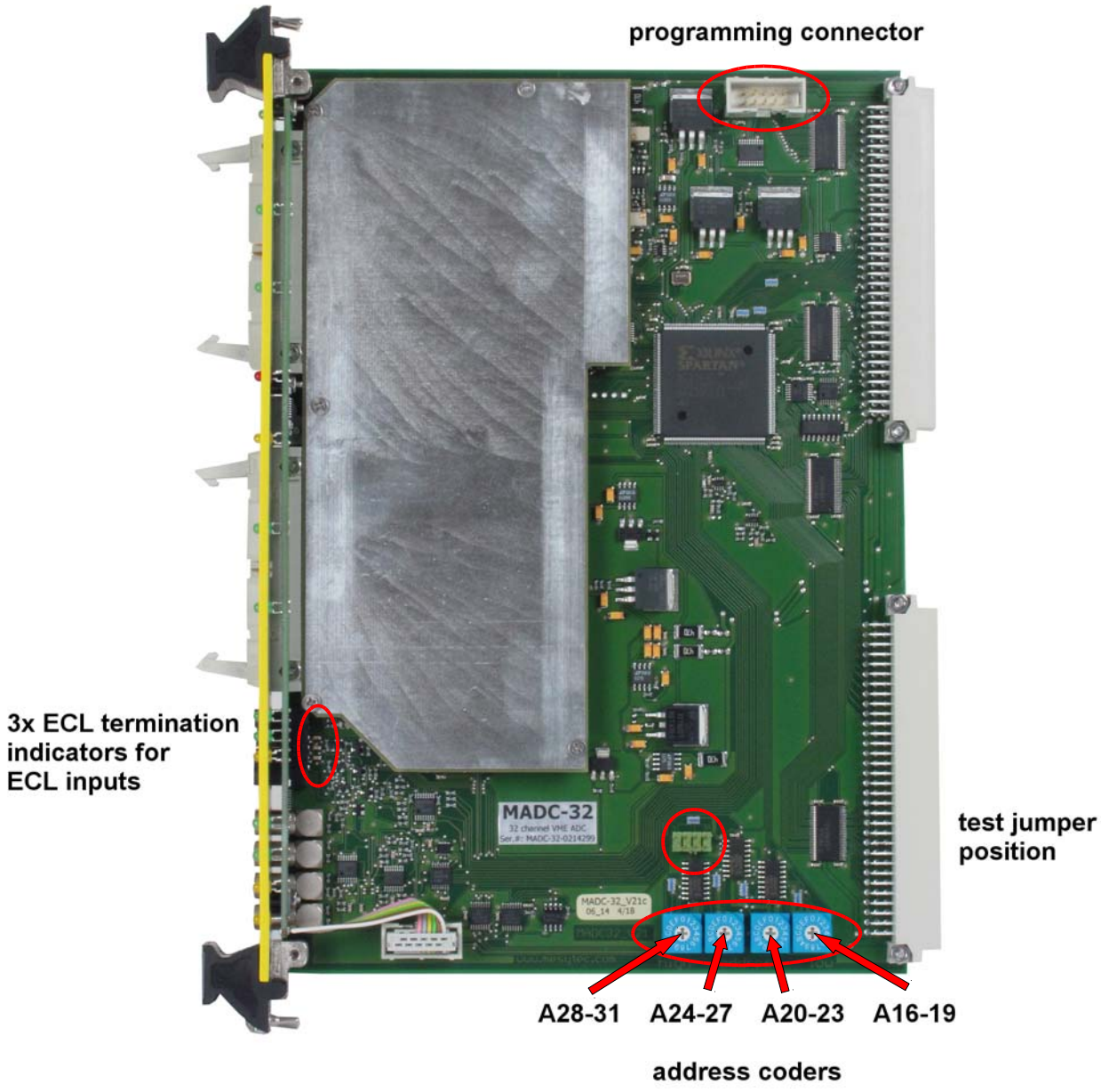
Using the toggle mode for 16 channels (very low dead time)

The data source (Shaper output) has to be connected to both input banks in parallel.
The gate0 input is used as trigger input, which means only a short pulse per conversion is needed.
The trigger pulses alternately start one of the two gate generators with their associated bank.
So while one gate generator or conversion is active in one bank, the other is ready for the next trigger.
The draw back is the reduction of channel number to only 16 for an MADC-32 module.

Fast clear

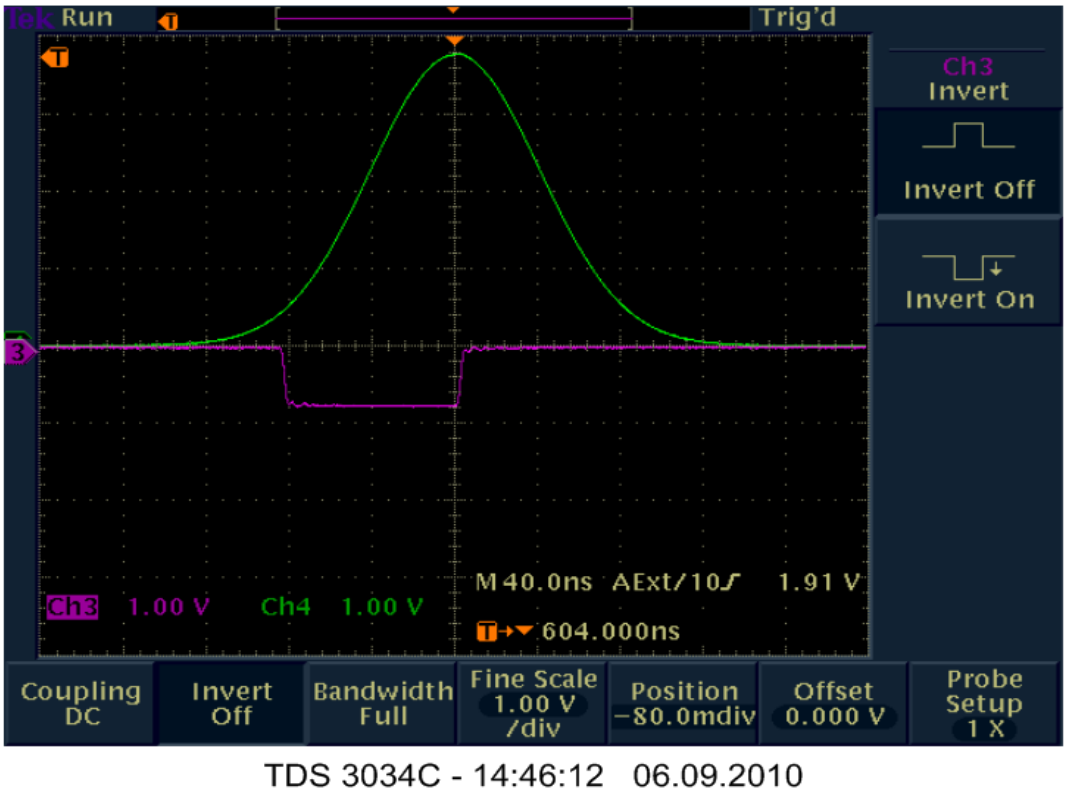
For some application it is useful to reject an event with some delay needed for trigger decision. The MADC-32 allows to clear an event from gate start to end of conversion (this can be 800 ns to 12.5us, depending on conversion time setting). Also when the built in gate generators are used the fast clear signal stops and resets them immediately. Fast clear time (from active edge of the fast clear signal **to next gate) is about 200 ns. The fast clear signal must be withdrawn before** next gate gets active.

MADC-32


programming connector
3x ECL termination indicators for ECL inputs
test jumper position
A28-31
A24-27
A20-23
A16-19
address coders

Appendix

Setting gates for 100 ns FWHM wide pulses. The gate should open at least 80 ns before the pulse maximum and may close at the maximum of the pulse.



Firmware Changes

FW0123 -	CBLT, BLT32, MBLT64 read access is always directed to the FIFO address 0 (regardless of the least significant 16 bit address)
FW0124 -	extended time stamp reset implemented
FW0125 -	improved VME bus filtering, decreases error rate at CMBLT64 and all other VMEbus operations
FW0126 -	Single shot reset for time stamp implemented (synchronisation with Xia modules)
FW0127 -	RC bus ready bit delayed by 100 us. Bus was not ready again at ready bit.
FW0200 -	mrc timing bug fix (bus not yet ready when ready bit goes low)
-	0x606e (RC-mode) setup needs up to 100 us. Read shows real status of busy/gate Ios
-	0x604C ignore thresholds implemented.
-	8 k words memory
-	extended time stamp allowed for independent bank operation
known bugs:	1) 0x603E misses first empty event (only 2 data words). work around: check 0x6030 > 0 2) VME read of time stamp register 0x609C, 0x609E may deliver wrong values.
FW0201 -	bugs fixed
FW0203 -	VME-interface updated (faster, better timing tolerance)
FW0220	Event counting in buffer, IRQ event threshold, allow to read out e fixed number of events Output of event threshold at NIM0 possible.