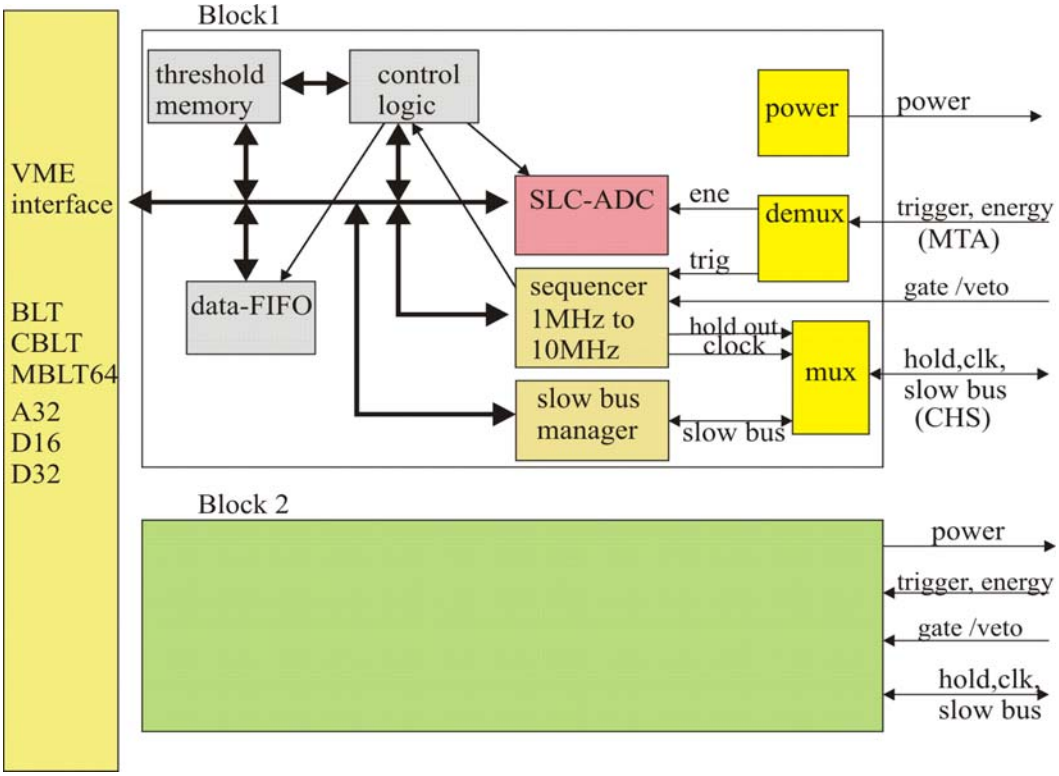
mesytec **MDI-2** is a VME sequencer and ADC for readout of MTM16 frontend modules. It provides 2 Buses, which allow to connect up to 16 MTM-16 frontend modules, and process their amplitude data. So up to 512 channels can be served by a single module.

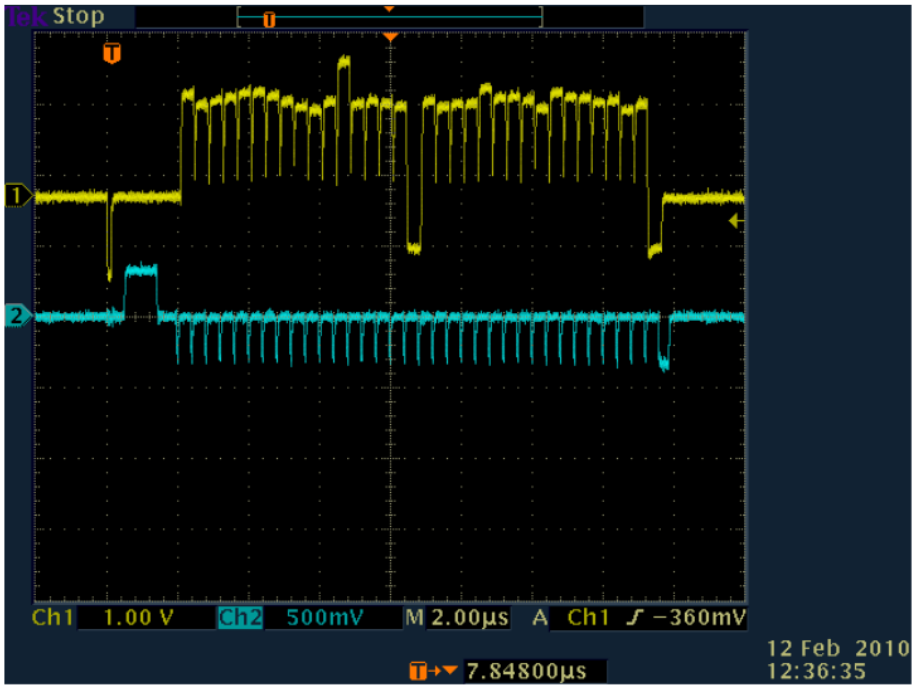
Features:

- High quality 12 bit (4 k) conversion with sliding scale ADC (DNL < 1 %)
- 10 M samples /s per bus.
- Up to 512 channels can be converted
- Multi event buffer
- Connected frontend modules can be remote controled (gain, threshold, polarity)
- 1 k words multi event buffer
- Zero supression with individual thresholds
- Supports different types of time stamping
- 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 timestamping start
- Easy compatibility to MADC-32 (only special registers 0x6040 to 0x6070 are different)

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

MDI2 Schematic:





Yellow MTA signal (monitor output). Blue CHS signal

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

Operation principle

For starting the readout of MTM-16 modules a trigger source is needed. It may come from an independent detector with strict timing correlation to the MTM-16 input event, or may come from some or all MTM-16 in one or several readout chains. MTM-16 trigger output can be inhibited for individual modules.
The individual trigger (multiplicity) outputs at MDI-2 are negative current outputs and can be connected, producing a current sum. For triggering external devices the common trigger output can be programmed to output trigger 0,1 or an ored trigger. If more complex trigger decisions are created by an external logic, the "veto" input can be used to accept or reject a trigger signal. The veto signal should be active before the rising edge of the internally created "gate" signal.
The "gate" delay and width, which is created inside MDI-2 for MTM-16 modules, can be adjusted via VME registers.
The trigger signal from MTM-16 is well suited as a high resolution timing signal.
The MDI-2 provides two TDC channels, one for each bus, which are started by the coresponding bus trigger and stopped by the stop inputs.
About 500 ns after the "gate" signal, the sequencer **MDI-2-sequencer** is started and produces the clock sequence for reading out the MTM-16 modules in two connected chains.
The incoming analog amplitude data are digitized by a **12 bit sliding scale ADC** (DNL <1 %) and compared with an individual **threshold** (one for each channel, configured via VME-bus). If the value is above threshold, it is stored together with the channel address in a memory (**fifo**).
After conversion ready the data can be accessed via VME bus.

MDI-2 also includes a control bus for each of the bus outputs which allow to communicate with the MTM-16 modules. It allows to set discriminator thresholds, amplification and polarity. An additional control bus is provided to control mesytec modules like Shapers and HV bias modules.

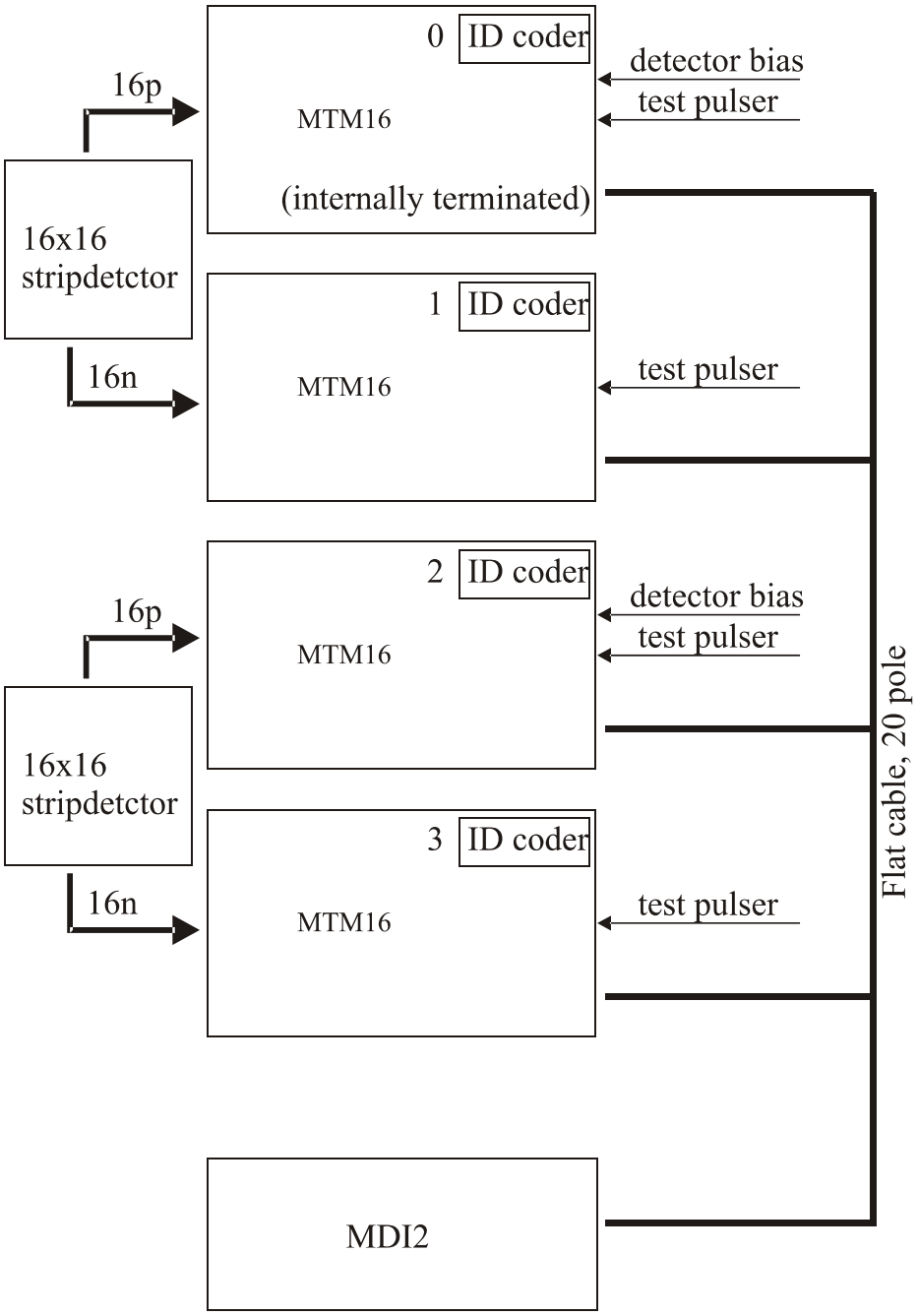
The MDI-2 is able to read out up to 2x16 MDI-16 modules, resulting in **512 channels**.

Front panel view of MDI-2

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

Chaining several MTM-16 Units

Chaining the MTM16 modules, connection to MDI2
Example: readout of two 16x16 silicon strip detectors



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

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

Input /output	direction	termination	Default functionality	Alternate functionalities
TRIG	In/Out	50 R	Common Trig-ger In/Out	-
VETO	Input	50 R	VETO	Gate (when active, triggers are accepted, see reg. 0x604C)
BUSY/ CBUS	In/Out	50 R	Busy out	1) Cbus I/O (0x6080) 2) External time stamp oscillator input (0x6096), max 20 MHz

Power consumption (Total: 4.5 W)

+ 5 V +230 mA
+12 V +160 mA
–12 V –100 mA

The power consumtion of connected MTM-16 mudules adds at +5 V and –12 V.

MDI-2 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	4 reserved	12 number of following data words, including EOE
b01	b000000	module id	b0000	number of 32 bit data words

Data (4 byte) DATA event

2 data-sig	4	10 Sample number	1 Bus number	1 out of range	2	12
b00	00 01	Sample number	0/1	Oor	b00	ADC amplitude

channel numbers may come in arbitrary order

Data (4 byte) Extended timestamp

2 data-sig	4	10	16
b00	00 11	00 1000 0000	16 high bits of timestamp

Data (4 byte), fill dummy (to fill MBLT-64 word at odd data number)

2 data-sig	4	10	1	1 -	2	12
b00	0	0	0	0	0	0

End of Event mark (4 byte)

2	30
b11	trigger counter / time stamp

Threshold memory at address x4000 to x4FFF (16 bit words, access: R/W D16)

Address	Name	Bits	dir	Default	Comment
0x4000	treshold_bus0[0]	12	RW	0	Threshold value of channel 0 value 0 = threshold not used
.	treshold_bus1[0]				
.	treshold_bus0[1]				
.	treshold_bus1[1]				
0x4008	treshold_bus0[2]	12	RW	0	Threshold value of bus0, channel 2

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).
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	10	RW	1	Every time the number of 32 bit words in the FIFO exceeds this threshold, an IRQ is emitted. Maximum allowed threshold is 956.
0x601A	Max_transfer_data	11	RW	1	Maximum data words to transfer before ending the transfer at next end of event word. Only works for multievent mode 3. At Max_transfer_data = 1, 1 event per transfer is emitted. Maximum number of events is 2047. Usually the same or higer value than in 0x6018 is used. Setting the value to 0 allows unlimited transfer.
0x601C	Withdraw IRQ	1	RW	1	Withdraw IRQ when data empty

For multievent 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 Enable MCST 0 No effect	0
6	MCSTDIS	1 Disable MCST 0 No effect	1 MCST enabled 0 MCST disabled
5	FIRSTENB	1 Enable first module in a CBLT chain 0 No effect	0
4	FIRSTDIS	1 Disable first module in a CBLT chain 0 No effect	1 First module in a CBLT chain 0 Not first module in a CBLT chain
3	LASTENB	1 Enable last module in an CBLT chain 0 No effect	0
2	LASTDIS	1 Disable last module in an CBLT chain 0 No effect	1 Last module in a CBLT chain 0 Not last module in a CBLT chain
1	CBLTENB	1 Enable CBLT 0 No effect	0
0	CBLTDIS	1 Disable CBLT 0 No effect	1 CBLT enabled 0 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 triggered, each Module emits the header and footer with time stamp (2 Words with 32 bits each: MDI-2 Header, MDI-2 footer).

	FIFO handling				
0x6030	buffer_data_length	14	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 (multievent **= 0**): allow new trigger, allow IRQ At multievent **= 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 mutievent **= 3** : clears Berr, allows next readout
0x6036	multievent	4	RW	0	allow multi event buffering (bit 0,1) **0 = no** (0x6034 clears event, allows new conversion) **1 = yes**, unlimited transfer, no readout reset required (0x6034 can be written after block readout) . Don't use for CBLT **3 = yes** but MDI 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 (instead of number of data words) with max_transfer_data (0x601A) for Berr condition.
0x6038	marking_type	2	RW	0	00 → event counter 01 → time stamp 11 → extended time stamp
0x603A	start_acq	1	RW	1	1 → start accepting gates If there is no external trigger logic, which stopps the triggers when daq is not runningis, 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

	trigger				
x6040	seq_enable	2	RW	3	enable sequecer 1,0 or both
x6042	trig_source0	3	RW	7	select trigger to start bus0 sequencer source: common, trigger1, trigger0

x6044	trig_source1	3	RW	7	select trigger to start bus0 sequencer source: common, trigger1, trigger 0
x6046	com_trig_source	2	RW	3	select source for common trigger I/O source: trigger1, trigger 0
x6048	gen_trigger	1	W		generate software trigger 0 / 1 (for test)
x604A	gen_event	1	W		generate trigger and event data for test. amplitude rising from 100 to 4 k (for test)
x604C	veto_gate	1	RW	0	0 = use "veto" input as **veto** veto is possible from trigger to 25 ns before end of hold-delay. 1 = use "veto" input as **gate** (when active triggers are accepted) veto and gate have to be active at least 25 ns before hold_delay0/1 runs out. If different delays for bus1/0 are used, the shorter one is relevant. (Typically 475 ns after trigger for MTM-16 timing)

	hold / gate generation				
x6050	hold_delay0	12	RW	1000	multiple of 0.5 ns (default: gate on bus0 starts 0.5 us after trigger)
x6052	hold_delay1	12	RW	1000	same for bus 1
x6054	hold_width0	7	RW	44	multiple of 25 ns (gate length on bus 0 default = 1.1 us)
x6056	hold_width1	7	RW	44	samefor bus 1

SQT 0x6060	**Sequencer timing**				
0x6060	bus_whatchdg	1	RW	1	bus whatchdog. 1 = on if no event trigger for 1 s, sends frontend reset. Protects against deadlocks.
0x6062	frontend_reset	0	W		frontend_reset (sends "reset sequencer" to MTM-16, settings left unchanged)
0x6064	seq_clk_freq0	3	RW	3	clock frequency, Bus 0: 0 = 1.25 MHz, 1 = 2.5 MHz, 2 = 5 MHz , 3 = 10 MHz
0x6066	seq_clk_freq1	3	RW	3	clock frequency, Bus 1
0x6068	seq_busy	2	R		read sequencer 1 / 0 busy when seq ready (= 0) , data are available at buffer
0x606A	sample_delay_reg0	4	RW	3	one additional tic delay for multiple of 2.5 m cables. For standard cable of 3 m set to 4
0x606C	sample_delay_reg1	4	RW	3	"
0x606E	enable_busy	1	RW	1	enable busy signal at Lemo output Do not allow when used as control bus or for timestamping

SQL 0x6070	Sequencer len				
0x6070	seq0_mct	4	RW	0	Not used
0x6072	seq1_mct	4	RW	0	Not used
0x6074	seq0_cct	10	RW	17	17 counts per MTM-16 needed
0x6076	seq1_cct	10	RW	17	"
0x6078	allow_sync_word	1	RW	0	allow_sync_words in fifo buffer. Useful when zero suppression in frontend MTM-16 (future feature)

Mesytec control bus

MRC 0x6080	**Module RC**				
0x6080	rc_busno	2	RW	0	0, 1 are the two readout buses 2 is the external bus, comes out at busy output
0x6082	rc_modnum	4	RW	0	0...15 (module Address set with hexcoder at MTM-16 modules)
0x6084	rc_opcode	7	RW		Comman codes sent to MTM-16: 3 = RC_on, 4 = RC_off, 6 = read_id, 16 = write_data, 18 = read_data
0x6086	rc_adr	8	RW		MTM-16 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 = any collision (no termination?) bit3 = no response from bus (no valid address)

Send time is 400µs. After this time response at 0x6088 is valid, and sending new data is possible.
At 0x608A bit0 = 1 while sending is active.

The Cbus-LED shows data traffic on the bus , the Cbus-Err-LED shows bus errors (i.e. non terminated lines)

Example for controlling MTM frontend modules connected to the buses

The MTM-16 has two registers (Address 0 and Address 1):

Data at Address 0

4	3	2	1	0
set rc-status 1 = set	rc_status (1 = on)	Not used	polarity 0 = neg. input 1 = pos input	gain 0 = high gain 1 = low gain

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

When set = 0, the bit values are not changed, but only read back.

Data at Address 1: threshold

12 Valid bits
Max value 0xfff = 50 % of single channel range

The data are immediately read back from the module and can be checked at register 0x6086 and 0x6088.
Eplicit reading with opcode 18 is not supported for MTM-16.

Example: set MTM with Address 5 at bus 1 to threshold to 0x0ff
(gets only active when RC is switched on)

```
write 0x6080 1          // bus 1
write 0x6082 5          // address module 5
write 0x6084 16         // decimal 16 = write to MTM-16
write 0x6086 1          // write to MTM-16 - address 1
write 0x6088 0x00ff     // set threshold to 0x00ff (about 3 % max range)
```

set MTM16 to positive, high gain and activate the remote control values
(deactivate G/P coder and threshold potentiometer)

```
write 0x6080 1          // bus 1
write 0x6080 5          // address module 5
write 0x6084 16         // decimal 16 = write to MTM-16
write 0x6086 0          // write to MTM-16 - address 0
write 0x6088 0b11010    // "set" "rc-status on" "nc" "positive pol." "high gain"
```

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 MDI-2 control bus at busy line
Write (16) addr 0x606E data 3

Get Module ID-Code (= Type of module = 20 for MSCF16)

```
Write (16)   addr 0x6080  data 2    // bus 2 (busy output)
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 bit 0 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 gain for channel 1 to 10

Write (16) addr 0x6080 data 2 // bus 2 (busy output)
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 0x6080 data 2 // bus 2 (busy output)
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 0x6080 data 2 // bus 2 (busy output)
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 MDI-2 control bus at busy line

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

CTRA

Timestamp 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	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	timestamp = 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	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		
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 1 k x 32 bit.
Data is organized in an event structure, maximum size of one event is 516 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				# of following words									
0	1	0	0	0	0	0	0	i	i	i	i	i	i	i	i	f	o	0	0	n	n	n	n	n	n	n	n	n	n	n	n

hsig: header signature = b01

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

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

output format f: currently = 0

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

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

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	c	c	c	c	c	c	c	c	c	c	b	v	0	0	d	d	d	d	d	d	d	d	d	d	d	d

dsig: data signature = b00

fix: bitfield currently without meaning = b0001

channel #: number of channel on bus (MTM-ID*16 + MTM channel address)
ADC channels may occur in arbitrary order

V: V = 1 indicates ADC overflow
b: bus number: 0 for bus0, 1 for bus1

ADC data: ADC conversion data, 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, MDI-2 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, MDI-2 emits a VME Berr (bus error)

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

Event structure when two buses are operated

The two independent sequencers can be started simultaneously with the same trigger or can be operated independently. When operated simultaneous, the data of both buses are merged into the same event structure (Header, data, end of event). When independently triggered, the two bus data are added into the same event when the time between the two triggers is less than 200 ns. When the difference is larger, the later trigger falls into conversion dead time and is skipped.

Data addresses

The Amplitude data are addressed in the order they are transmitted by the buses.
As the MTM-16 does not transmit the amplitude in the straight channel order, the addresses have to be assigned to channel numbers by the user.

Channel Order by MTM-16	Sample number in MDI-2 event structure
0	0
8	1
1	2
9	3
2	4
10	5
3	6
11	7
4	8
12	9
5	10
13	11
6	12
14	13
7	14
15	15

The MDI-2 read out in two modes

Single event readout

In this mode the event data from buses 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 gtrigger to start the conversion. Triggers coming within the time from accepted trigger 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 lengt
   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

Hardware configuration:
- 1x MTM-16 25 MeV connected with 1.5 m twisted pair cable to Bus0 of MDI-2
- MTM frontend: Address set to 0
- MTM frontend: R/P switch set to E
- Tail-pulser applied to pulser input, externally terminated pulser line, amplitude –200 mV, 50 ns rise time, 100 us decay time

MDI sequencer setup (default setting)

Reg 0x6050 → 1000 (gate0 delay 500 ns)
Reg 0x6054 → 44 (gate0 length 1.1 us)
Reg 0x6074 → 17 (16 data + 1x Sync)
Reg 0x606A → 3 (delay adapted to 1.5 m of cable)

Stop acquisition: 0x603A = 0; Stop
Set multievent 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 lengt (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 multievent readout mode (0x6036, mutievent = 1 or 3) the input is decoupled from output by a 1024 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 methode is **event counting**.
Each module has an event counter and counts the incomming 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 MDI-2 this can be the VME built in clock of 16 MHz or an external clock up to 20 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 incomming events are then labeled with a 30 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 methodes allow the different modules to be completely independent from each other. It gets now possible to use large data buffers in the frontend modules, and do the readout when the VME data bus is not occupied. The MDI-2 alows 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 three possibilities.

1. multievent mode = 1: read “buffer_data_length” (0x6030) and transfer the amount of data read there.

2. multievent 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 incoming data rate on both buses may be as high as 20 Mwords/s, the VME readout rate is realistic about 5 Mwords /s in BLT32, 10 Mwords/s at MBLT64. 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. multievent mode = 3 : MDI transfers a 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).

IRQ

For many setups it is useful to control the readout via interrup requests (IRQ) defined by VME.
For MDI-2 an IRQ is initiated when the buffer fill level gets above the “irq_threshold” (0x6018).
The IRQ is acknowledged by the VME controler, then the controler starts a readout sequence.
When not using the readout reset (0x6034) at the end of a readout cycle, the MDI 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 MDI modules in VME bin this can also be done with a single multicast write.

Example 1, multievent 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 timestamp in EOE mark	marking type	0x6038 = 1;
Synchronisation:	reset_ctr_ab	0x6090 = 3; reset all counters

3. Set Multievent

Multievent 0x6036 = 3	Multievent 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 = 218	Transmit maximum 223 words + reset 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 or MBLT64) until BERR on VME-bus.
Then write reset register 0x6034 (D16)

Example 2, chained block transfer

Describes multi event readout but with 2 MDCs and 1 MDI-2 via 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:
MADC 1 in slot 1 gets 0x0100
MDI 2 in slot 2 gets 0x0200

MADC 3 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)
MDI2: 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 multievent mode 0 or 3 for CBLT (mode 1 will not work !)

Special Operation

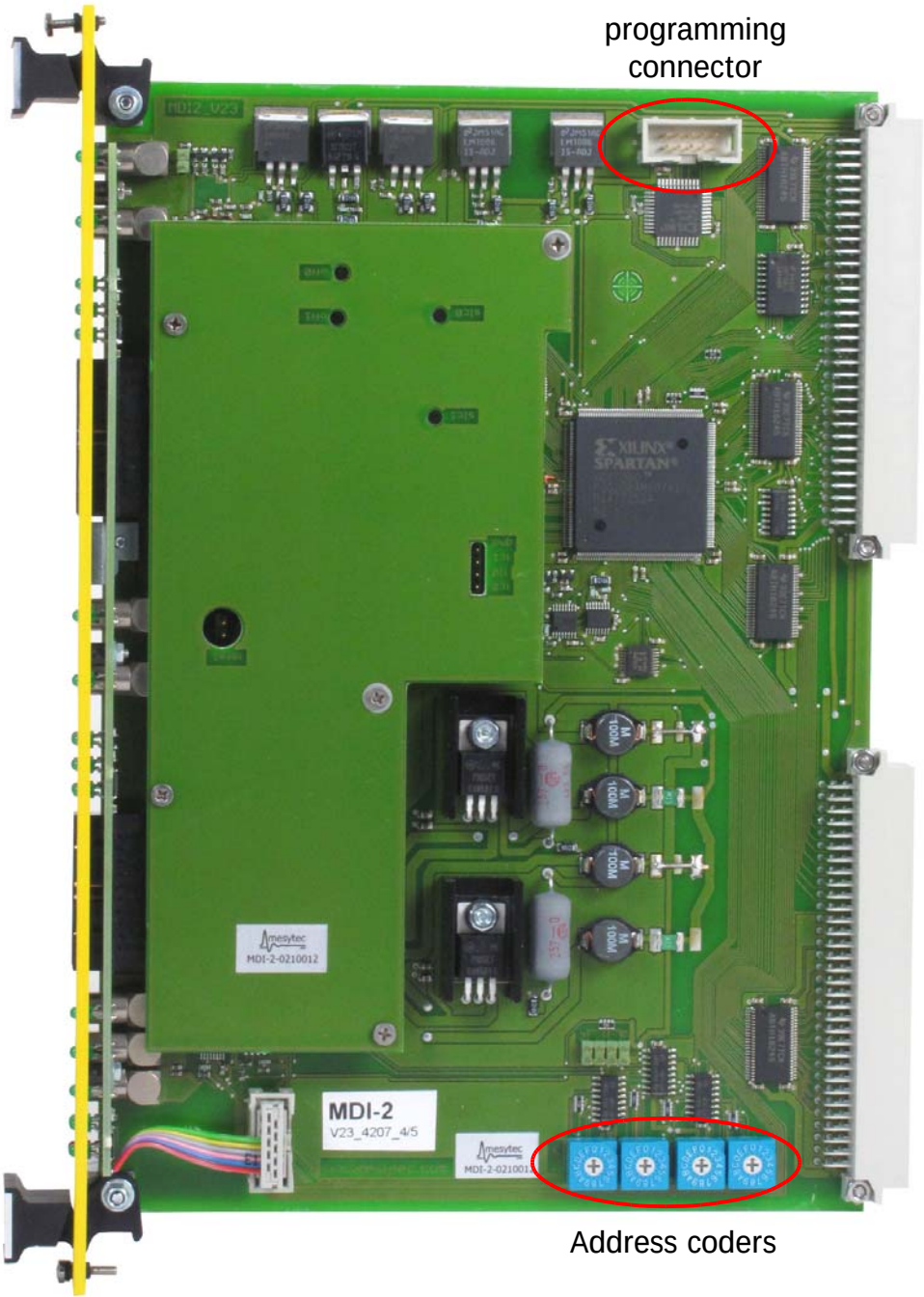
MBLT64

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 !

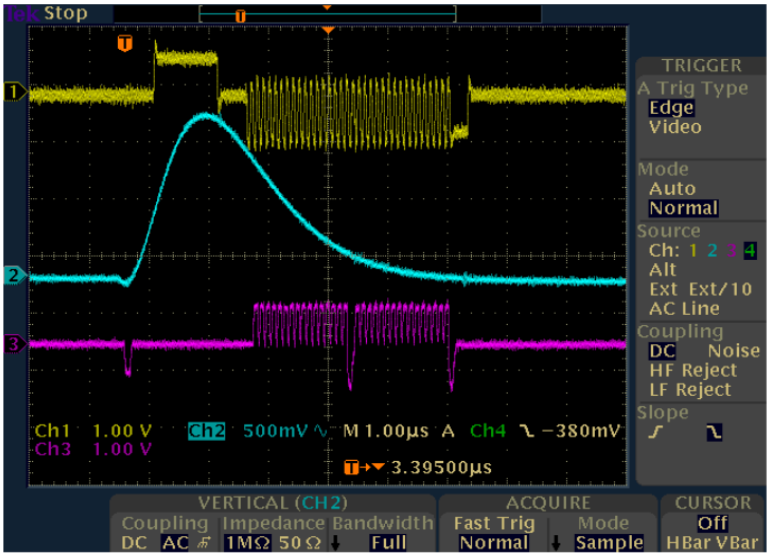
MDI2 overview



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

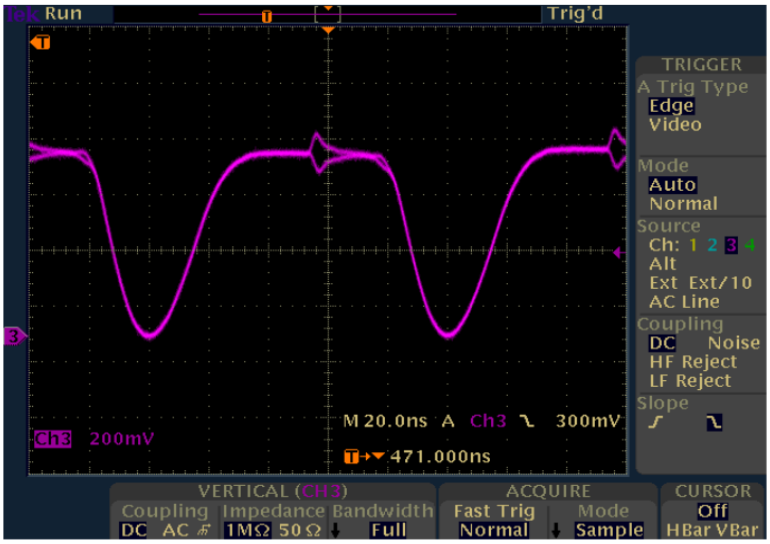
Scope traces showing timing structure on the bus

MDI-2 with two MTM-16 connected at bus0, 10 MHz sequencer frequency



2x MTM-16 read out with MDI-2

Yellow: CHS-signal: in order of occurrence: Gate, clocking sequence, reset pulse
Blue: Shaper output of one channel inside MTM-16
Magenta: MDI-2 monitor signal, available at front panel. In order of occurrence:
Trigger pulse, 32 Amplitude values with 2 negative synchronisation pulses



MTM-16 readout

Monitor signal. Flicker mark maks the ADC sampling time. The flicker position can be adjusted by register 0x6054 / 0x6056 (in steps of 25 ns) and by the bus cable length.
Adding 1 m cable to the bus, shifts the sampling time 10 ns to the left.
The sampling time should be about 10 ns before the falling edge of the amplitude signal.

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

Power supply of MTM-16 modules

The MTM-16 modules are supplied with power via their bus line.
The length of the bus line is limited by the voltage drop. Typical 12 mV/m/module can be expected.
So for 4 MTM-16 modules with a bus length of 3 m a voltage drop of 4 x 4 x 12 mV = 192 mV can be expected. The maximum allowed drop is 400 mV. If the voltage drop exceeds this value, the modules need an external supply.

Several MTM-16 modules in a chain

When several modules are connected to a bus, the Address coders must be set from 0 to module number -1.
For example: 5 modules at bus 0: the coders are set to 0, 1, 2, 3, 4 or 1, 0, 4, 3, 2.
Not allowed is for example 1, 2, 3, 4, 5 or 0, 2, 3, 4, 5.

The bus line must be terminated with a special terminator plug.
It terminates the CHS and MTA lines with 50 Ω to ground.