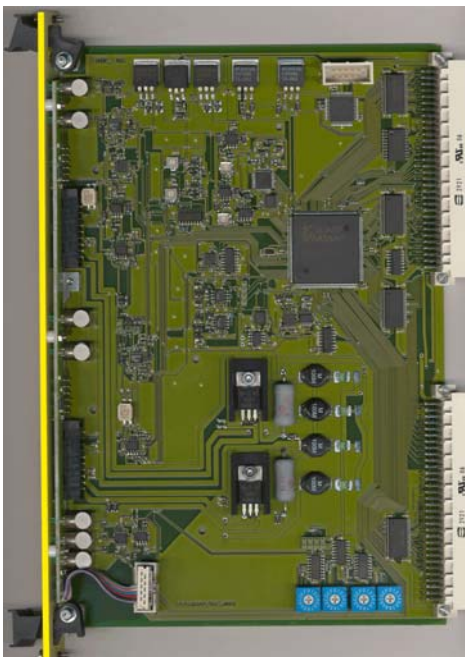


High resolution with low differential non linearity (sliding scale ADCs).  
12 Bit (4k).  
Up to 10M Samples per second per bus.

Supports zero suppression with individual thresholds in linear transmission mode.  
Supports MTM16 zero suppression mode.  
One multiplicity current output per bus.

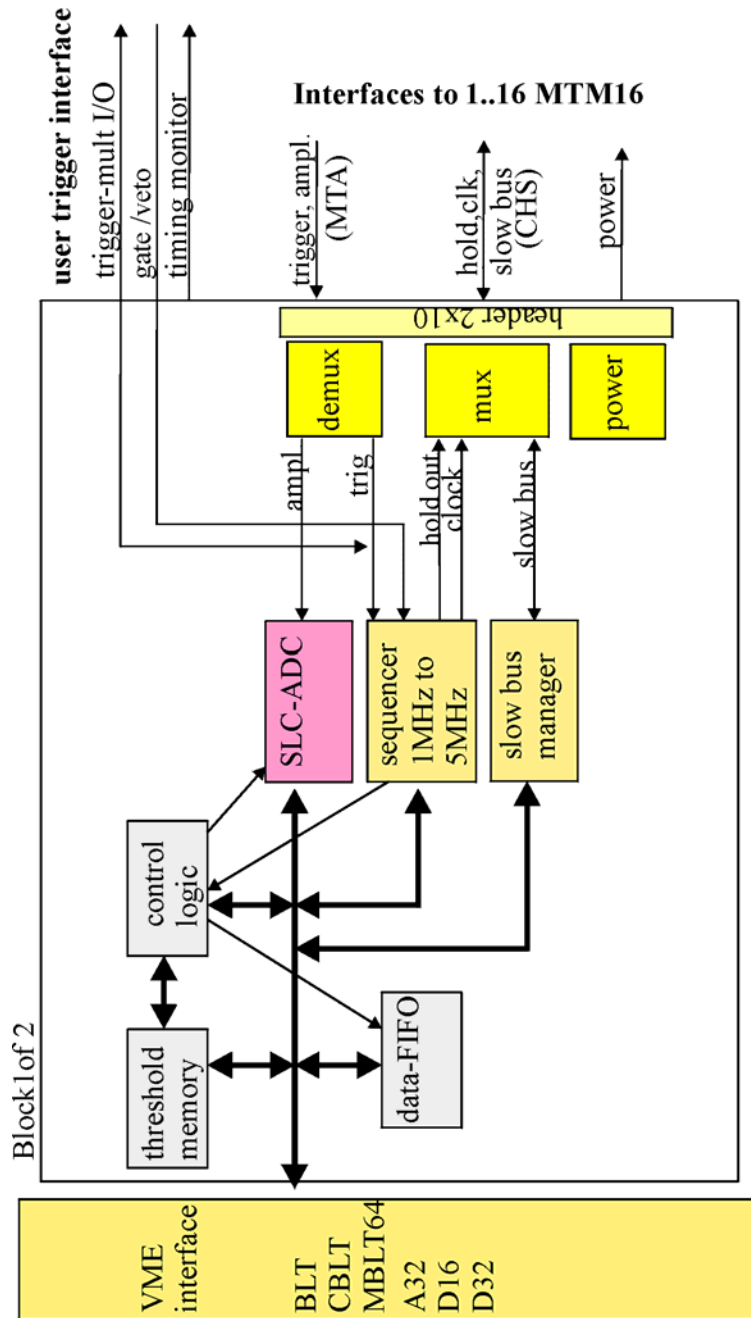
### Features:

- High quality 12 bit (4k) conversion with sliding scale ADC (DNL < 1%).
- 10 M samples / s per bus.
- Up to 512 channels can be converted
- Multi event buffer
- Zero suppression with individual thresholds
- Supports different types of time stamping
- Connected frontend modules can be remote controlled (gain, threshold, polarity).
- Address modes: A24 / A32
- Data transfer modes: D16,32,64, BLT32, MBLT64, CBLT
- Multicast for event reset and timestamping start.



# MDI2 Schematic

## MDI2



## Operation principle:

For starting the readout of MTM16 modules a trigger source is needed. It may come from an independent detector with strict timing correlation to the MTM16 input event, or may come from some or all MTM16 in one or several readout chains. MTM16 trigger output can be inhibited for individual modules.

The individual trigger (multiplicity) outputs at MDI2 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 (1 us after trigger).

The "gate" delay and width, which is created inside MDI2 for MTM16 modules, can be adjusted via VME registers.

The trigger signal from MTM16 is well suited as a high resolution timing signal.

The MDI2 provides two TDC channels which are started by the trigger and stopped by the stop inputs. About 500 ns after the "gate" signal, the sequencer **MDI2-sequencer** is started and produces the clock sequence for reading out the MTM16 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 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.

MDI2 also includes a control bus for each of the bus outputs which allow to communicate with the MTM16 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.

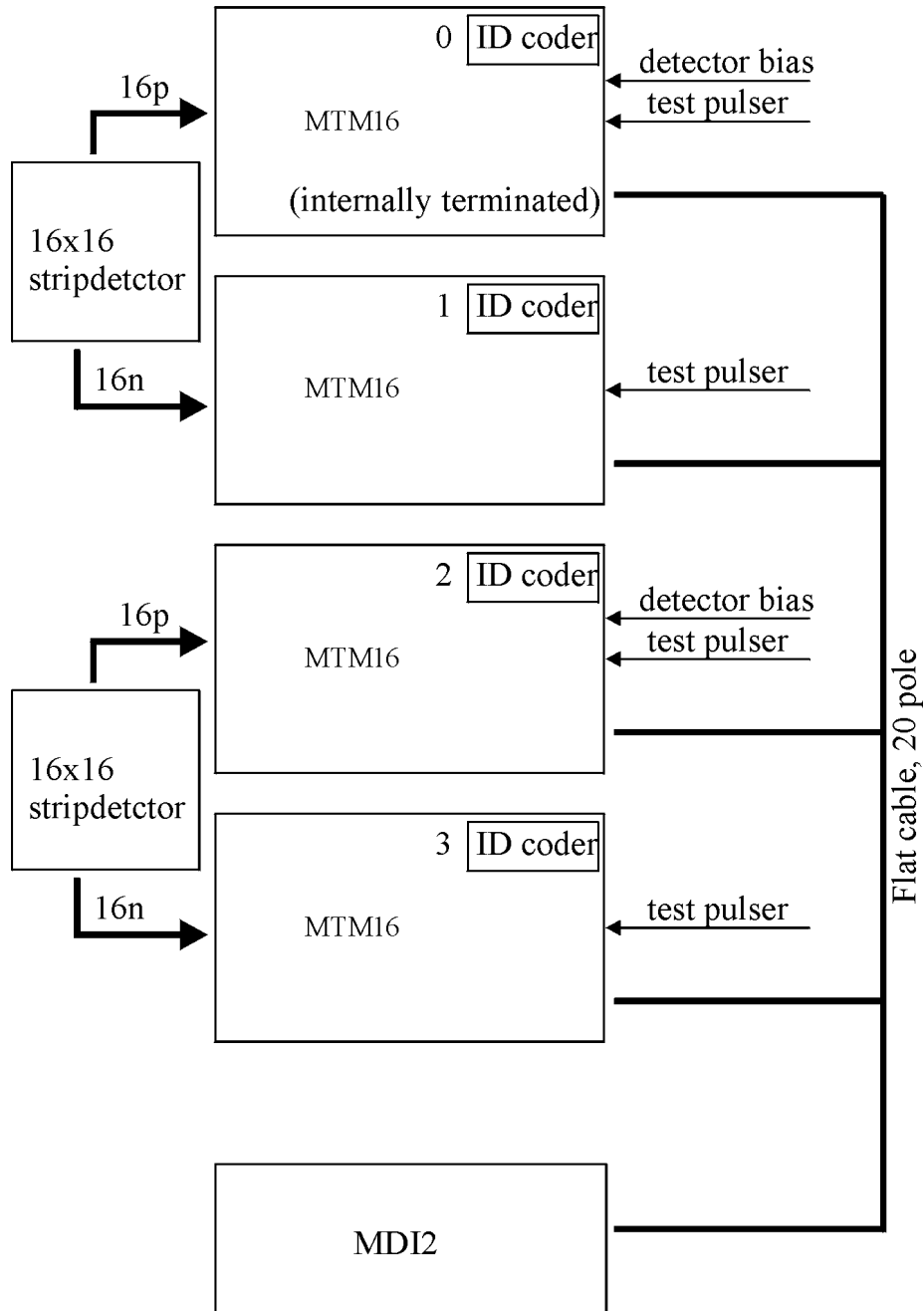
The MDI2 is able to read out up to 2x16 MDI16 modules, resulting in **512 channels**. Future frontend modules will allow to connect up to 2000 channels.

## Front panel view of MDI2



## Chaining several MTM16 Units

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



## MDI2 register set.

Items marked in blue are not implemented in the actual firmware revision

### Data memory, starting at address x0000 (access R/W D16,32,64)

for 64 bit access, last 32bits will be filled with 0 for odd number of 32 bit data words  
memory size: 1k x 32

#### Header (4 byte)

2	6	8	4	1	11
header signature	subheader	module id	reserved	buffer overflow	number of following data words, including EOE
b01	b000000	module id	b000	event_err	number of 32 bit data words

#### Data (4 byte) DATA-event

2	4	10	1	1	2	12
data-sig				overflow		
b00	b0001	bus sample no	.bus no	overflow	b00	ADC amplitude

sample no. + bus no. may come in arbitrary order, depending on sequencer frequencies

#### Data (4 byte) SYNC-event

2	4	10	1	1	2	12
data-sig				overflow		
b00	b0011	bus sample no	.bus no	0	b00	0

#### Data (4 byte) TDC-event

2	4	10	1	2	1	12
data-sig				-	overflow	
b00	b0010	0	bus no	b00	overflow	TDC value

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

2	4	10	1	2	1	12
data-sig				-	overflow	
b00	b0000	0	0	0	0	0

#### End of Event mark (4 byte)

2	30
b11	trigger counter / time stamp

At read of EOE, buffers and sequencers are resetted, the module waits for next trigger.

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

Address	Name	Bits	dir	default	Comment
x4000	treshold_bus0[0]	12	RW	0	Threshold value of bus0, channel 0 value 0 = threshold not used
x4002	treshold_bus1[0]	12	RW	0	
x4004	treshold_bus0[1]	12	RW	0	
x4006	treshold_bus1[1]	12	RW	0	
x4008	treshold_bus0[2]	12	RW	0	Threshold value of bus0, channel 2

1k thresholds per bus = 2k x 16.

### Registers, Starting at address x6000 (access D16)

Address	Name	Bits	dir	default	Comment
	<b>Address registers</b>				
x6000	address_source	1	RW	0	0 = from board coder, 1 from address_reg
x6002	address_reg	16	RW	0	address to override decoder on board
x6004	module_id	8	RW	1	is part of data header
x6008	soft_reset	1	W		breaks all activities, sets critical parameters to default
x600A					
x600C					
x600E					
	<b>IRQ</b>				
x6010	irq_level	3	RW	0	IRQ priority 1..7, 0 = IRQ off
x6012	irq_vector	8	RW	0	IRQ return value
x6014	irq_test	0	W		initiates an IRQ
x6016	irq_reset	0	W		resets IRQ
x6018					
x601A					
x601C					
x601E					
	<b>MCST CBLT</b>				
x6020	cbt_control	2	RW	0	passive = 0, first, mid, last
x6022	mcst_cblt_address	8	RW	AA	A24 high address
x6024	berr_flag	1	R		for BLT CBLT, resets when read
x6026	mcst_cblt_reset		W		
x6028					
x602A					
x602C					
x602E					
	<b>FIFO handling</b>				
x6030	buffer_data_length	14	R		amount of data in FIFO (only fully converted events). Units -> data_len_format. converted from actual event header. Only makes sense in single event buffering.
x6032	data_len_format	2	RW	2	0= 8bit, 1=16bit, 2=32bit, 3=64bit At 3 a fill word may be added to the buffer to get even number of 32 bit words.
x6034	readout_reset		W		clears all ready converted events. At single event mode: allow new trigger, allow IRQ
x6036	multievent	1	RW	0	allow multi event buffering (0=no)
x6038	marking_type	1	RW	0	event counter (=0) or time stamp
x603A	start_acq	1	RW	1	start (=1) or stop acquisition
x603C	fifo_reset		W		initialise fifo
x603E	data_ready	1	R		data available

	<b>trigger</b>				
x6040	seq_enable	2	RW	3	enable sequecer 1,0 or both
x6042	trig_source0	3	RW	7	source: common, trigger1, trigger0 (source to start seqencer 0 on bus 0)
x6044	trig_source1	3	RW	7	source: common, trigger1, trigger0
x6046	com_trig_source	2	RW	3	source: trigger1, trigger0 (feeds common trigger I/O)
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 4k (for test)
x604C	veto_gate	1	RW	0	0= use "veto" input as veto, 1= use "veto" input as gate veto and gate have to be active at least 25ns before hold_delay0/1 runs out. If different delays for bus1/0 are used, the shorter one is relevant. (Typically 450ns after trigger for MTM16 timing)
	<b>hold / gate gen- eration</b>				
x6050	hold_delay0	12	RW	1000	multiple of 1ns (gate on bus0 starts 0.5us after trigger)
x6052	hold_delay1	12	RW	1000	same for bus 1
x6054	hold_width0	7	RW	34	multiple of 25ns (gate length on bus 0 default =850ns
x6056	hold_width1	7	RW	34	samefor bus 1
<b>SQT # 6060</b>	<b>Sequencer timing</b>				
SQT+0	bus_whatchdg	1	RW	1	bus whatchdog. 1= on if no event trigger for 1s, sends frontend reset. Protects against deadlocks.
SQT+2	frontend_reset	0	W		frontend_reset (sends "reset sequencer" to MTM16, settings left unchanged)
SQT+4	seq_clk_freq0	3	RW	3	clock frequency:0=1.25, 2.5, 5 ,3=10MHz
SQT+6	seq_clk_freq1	3	RW	2	clock frequency:0=1.25, 2.5, 5 ,3=10MHz
SQT+8	seq_busy	2	R		read sequencer 1 / 0 busy when seq ready (=0) , data are available at buffer
SQT+A	sample_delay_reg0	4	RW	3	one additional tic delay for multiple of 2.5m cables. For standard cable of 3m set to 4
SQT+C	sample_delay_reg1	4	RW	3	"
SQT+E	enable_busy	1	RW	1	enable busy signal at Lemo output Do not allow when used as control bus or for timestamping
<b>SQL #6070</b>	<b>Sequencer len</b>				
SQL +0	seq0_mct	4	RW	0	module count (0= off)

SQL +2	seq1_mct	4	RW	0	
SQL +4	seq0_cct	10	RW	17	17 counts per MTM16 needed
SQL +6	seq1_cct	10	RW	17	"
SQL +8	allow_sync_word	1	RW	0	allow_sync_words in fifo buffer. Useful when zero suppression in frontend MTM16 (future feature)

<b>MRC #6080</b>	<b>Module RC</b>				
MRC+0	rc_busno	2	RW	0	0,1 local buses, 2 is external bus, comes out at busy output
MRC+2	rc_modnum	4	RW	0	0...15 (module ID set with hexcoder at MTM16)
MRC+4	rc_opcode	7	RW		3=RC_on, 4=RC_off, 6=read_id, 16=write_data, 18=read_data
MRC+6	rc_adr	8	RW		module internal address, see box below
MRC+8	rc_dat	16	RW		data (send or receive), write starts sending
MRC+A	send return status	4	R		bit0=active bit1=address collision bit2=any collision (no termination?) bit3=no response from MTM16

**MTM16 commands: (MTM can be initialised with 2 write commands)**

**Address 0:**

4	3	2	1	0
set rc-satus 1= set	rc_satus (1= on)	deact. trigger	polarity 0= neg. input 1= pos input	gain 0= high gain 1= low gain

**Address 1: set common threshold**

dat[11:0] -> dac (0xfff = 50% of single channel range)

The data are immediately read back from the module and can be checked at register 0x6086 and 0x6088. Explicit reading with opcode 18 is not supported for present MTM16.

Send time is 400us. Wait that fixed time before reading response or sending new data. Also polling at 0x608A for bit 0 is possible

<b>CTRA1 #6090</b>	<b>countersA</b>				
CTRA+0	evctr_res		W		
CTRA+2	evctr_lo	16	R	0	event counter low value
CTRA+4	evctr_hi	16	R	0	event counter high value
CTRA+6	ts_osc_source	1	RW	0	frequency source (VME=0, external=1)
CTRA+8	ts_reset_source	1	RW	0	external reset enable = 1

CTRA+A	ts_reset		W		usually MCST
CTRA+C	ts_counter_lo	16	R		time stamp counter
CTRA+E	ts_counter_hi	16	R		time stamp counter
<b>CTRB #60A0</b>	<b>countersB</b>				
CTRB+0	seq_time_lo	16	R		seq active time, from gate to end of sequence [10us]
CTRB+2	seq_time_hi	16	R		
CTRB+4	daq_time_lo	16	R		from VME interrupt to eor_reset [10us]
CTRB+6	daq_time_hi	16	R		
CTRB+8	time_0	16	R		time [1us] (48 bit)
CTRB+A	time_1	16	R		
CTRB+C	time_2	16	R		
CTRB+E	time_reset		W		reset all 3 time counters
	<b>TDC</b>				
x60C0	tdc_active0	1	RW	0	1 activates TDC0 start input, deactivates mon0 output
x60C0	tdc_active1	1	RW	0	1 activates TDC1 start input, deactivates mon1 output

For internal use only

	<b>Fifo pointers</b>				
x60F0	buffer_start_ptr	10	R		
x60F2	buffer_read_ptr	10	R		
x60F4	data_end_ptr	10	R		
x60F6	seq_header_ptr	10	R		
x60F8	seq_data_ptr	10	R		

## Signal lines

### MDI2 interface to MTM16 PCBs:

20 lines, twisted pair required for long distances

1,2	+12V
3,4,5	+5V
6,8	-5V
7,9,10,11,12	power gnd
13	+CHS
14	- CHS

15	overtemp (current, wired or) OPTION
16	analog gnd
17	+MTA
18	-MTA
19, 20	analog gnd

# Data handling

The event buffer is organised as a FIFO with a depth of 1k x 32bit. One converted amplitude occupies a 32 bit word.

The data from the frontend modules are digitized and stored in the following structure in the event buffer:

## Header (4 byte)

2	6	8	4	1	11
header signature	subheader	module id	reserved	buffer overflow	number of following data words, including EOE
b01	b000000	module id	b000	event_err	number of 32 bit data words

The highest two bits define the data frame: 01 for header, 00 for data, 11 for the end of event word. The subheader field is used to define different data types for words marked as data (00).

A unique module ID (may code for crate\_number and Slot\_number) allows to determine the source module.

The buffer overflow bit is set when not all data could be written to the buffer. The word number in the header and EOE marker will be correct anyway.

The word\_number field

## Data (4 byte)

2	4	10	1	1	2	12
data-sig				overflow		
b00	b0001	bus sample no	.bus no	overflow	b00	ADC amplitude

sample no. + bus no. may come in arbitrary order, depending on sequencer frequencies

## Data (4 byte)

2	4	10	1	2	1	12
data-sig				-	overflow	
b00	b0010	0	bus no	b00	0	TDC value

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

2	4	10	1	2	1	12
data-sig				-	overflow	
b00	b0000	0	0	0	0	0

## End of Event mark (4 byte)

2	30
b11	trigger counter / time stamp

At reading beyond EOE, a VME Berr (bus error) is emitted. It can be used to break block transfer or multi block transfer.

Events can be read out in two modes:

**Single event readout** (only implemented at the moment, multievent readout will come soon)

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

Readout of a single MTM16.

Set MTM Address coder to 0

Set MTM gain/pol coder to "E" (high range positive pulser input)

Apply positive tail pulser, risetime 100ns, decay time 100us

Remote Controlling of MTM16 is not necessary for a first test.

All sequencer data can be left at default.

All trigger data can be left at default

Initialise IRQ:

set IRQ: set reg 0x6010 to 1 (IRQ-1 will be set when event is converted)

set reg 0x6010 to 0 (IRQ Vector)

Reset fifo:

write register 0x6034 (any value)

Now module is ready for IRQ triggered readout loop:

Read register 0x6030 for event length (D16)

Read from buffer event\_length + 1 (BLT32)

Write reset register 0x6034 (D16)

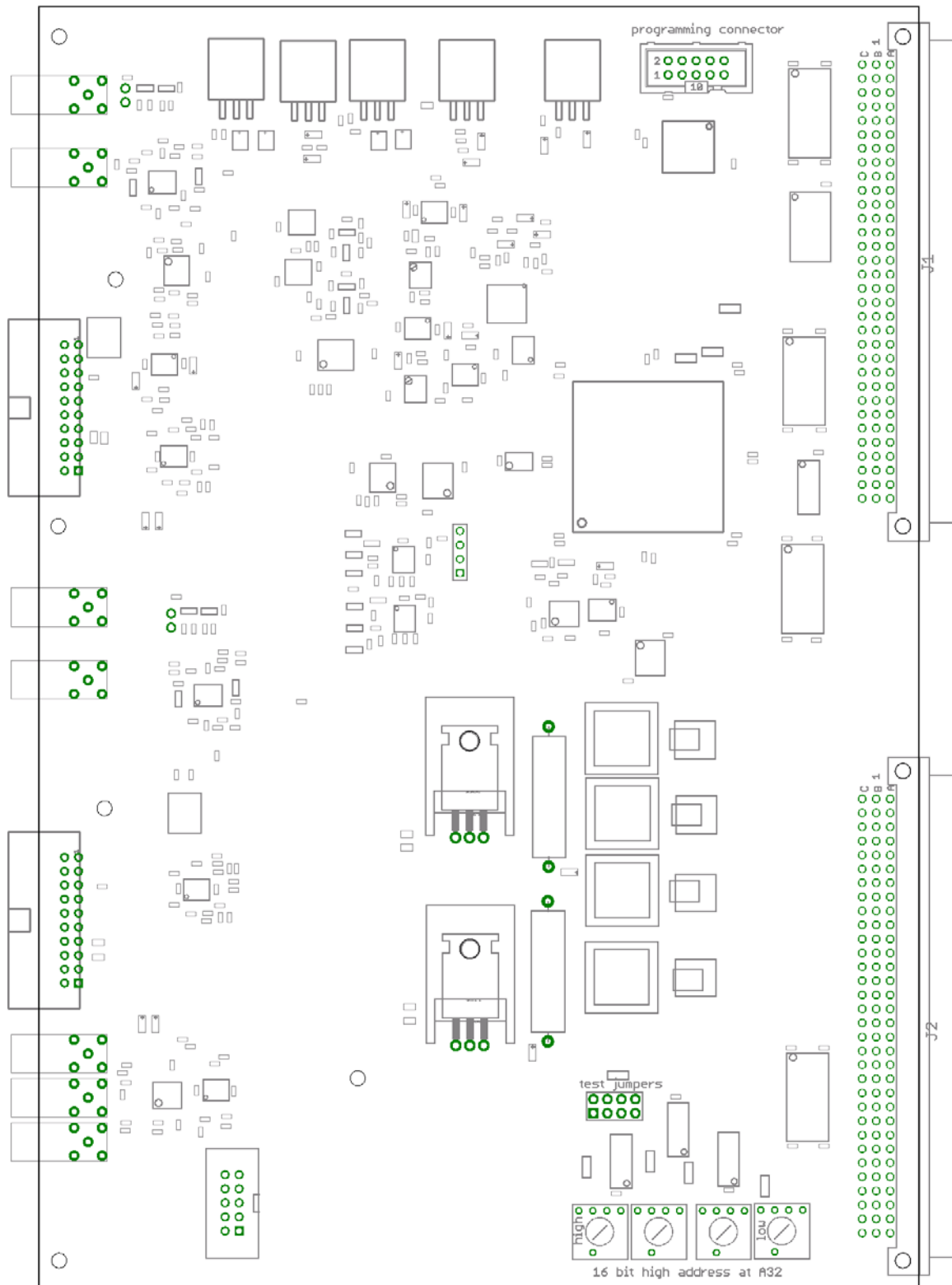
Or:

Start block transfer (BLT32) until BERR on VME-bus

Then write reset register 0x6034 (D16)

## MDI2

Address coders,  
programming connector  
test jumper position



## Revisions

FW 1.0.1 : Endianess Bug fixed

FW 1.0.2 :Veto input can be used as gate input. Default Veto (change at reg 0x604C, 1 -> Gate)

FW 1.0.3 : proper limits for ADC underflow and overflow implemented. busy default (reg 0x606E) -> on