# ADC board for BOREXINO experiment

# DRAFT SPECIFICATIONS

Paolo Musico, Armida Nostro

*INFN Genova - via Dodecaneso, 33 - 16146 - GENOVA - Italy*

*16 April 1998*

## *Abstract*

In this document we define the technical specifications of the multichannel multisampling analog to digital converter board, which can be used for the events energy measurement in the Borexino experiment.
This unit will be designed to be as general as possible, so it could be used also for different purposes and in different configurations.
A detailed description of the all board functions is given, together with a list for the foreseen active components.
A cost estimation is also given, keeping into account components and PCB procurements, assembly and testing.

## *Requirements*

We have to keep in consideration the requirements coming from the experiment. The requirements for the design of this unit are the following:
Electrical and mechanical specifications conform to VME standard

| | |
|---|---|
| Input voltage range: | 2Vpp |
| Input signals shape: | quasi trapezoidal, with about 20 nsec rise time and about 100 nsec width |
| Rate of input signals: | 20 KHz mean, with peak rate at 100-200 KHz |
| Resolution: | 10 bit at least |
| Sampling frequency: | 40 MHz or greater |

For each channel there is a corresponding differential ECL signal (coming from an external discriminator stage) in time with the rising edge of the analog input.
The board, for each channel, must be able to store at least one sample before the discriminator signal and at least three samples after.
At the same time of the discriminator firing, we need to store also a time tag, used to associate the data with the **trigger**.
For each channel the board must store all data from 1 μsec before to several μsec after the **trigger** signal (coming from external logic).
Together the data the board must store also the time difference between data and trigger, and a trigger ID number.
The trigger latency is in the order of 1 μsec.
The acquisition time (acceptance window) is nominally of 5 μsec (1 μsec before, and 4 μsec after), but we need to be ready to acquire data also for 100 μsec or more.

The board should also use converted data (non triggered) for statistical measurements and transfer them to the crate CPU for system slow control purposes.

## *Implementation*

Electrical and mechanical specifications conform to VME standard 9U x 400 mm, with power supply conform to CERN v430 standard, using Jaux connector for ECL power supply and geographical addressing.
On the board we can house 32 independent A/D conversion channels and the common readout and control logic, together with VME interface and memory buffers.

The possibility to group 4 channels together, feeded by the same inputs via an external fanout, and forming a 4 times faster digitizer is foreseen. In this case (software programmable option) an 8 channels board running at 160 MHz is implemented (if nominal channel frequency is 40 MHz).

The main controller of the unit will be a DSP from Texas Instruments, model TMS320C50, to guarantee the maximum flexibility in the channels readout management, in the data formatting and in the statistical analysis.

At writing time the components cost prevision is about 150.000 Lit/channel excluding VAT.
For the Borexino full production (80-100 units), keeping into account a multiplying factor of 2.6, to include PCB, assembly and testing cost, the channel cost is foreseen to be about 400.000 Lit/channel (always excluding VAT).

## A/D conversion channel

The measurement logic will be to sample N times the input signal, starting from the discriminator input, at 40 MHz frequency, with the first sample placed before the rising edge.
Doing this way it is possible not only measure the peak value, but also reconstruct the input signal shape (if we know the transfer function of the preceding stage, of course) and evaluate the possible pile-up introduced by near events or low frequency noise.
We foresee to run the AD converter at 40 MHz rate and to store 4 samples of the input signals.
The possibility to store data before the discriminator signal is implemented using the internal pipeline of the sampling

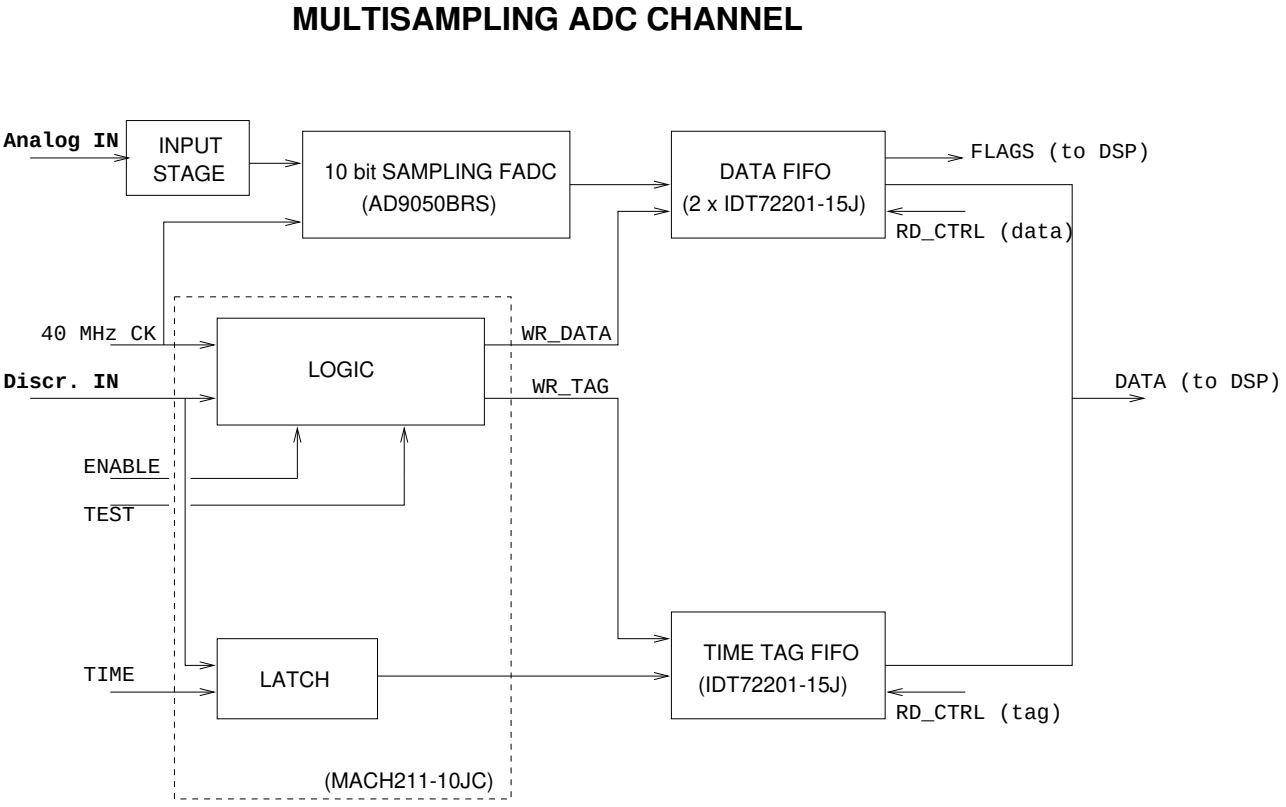


**Fig. 1: The ADC channel**

AD converter components.
The stage is designed to be as flexible as possible, in order to be used in different configurations.
The block diagram of the AD channel is presented in Fig. 1.
The main component is the Analog to Digital converter which is free running at 40 MHz clock rate, preceded by an analog stage used to attenuate or amplificate and translate the input signal.
The coupling of the input signal is in AC, but it is possible also CC coupling, shorting the input capacitor.
A protection stage against over-under voltage is foreseen. If some application do not need it, is possible to remove it without any implication.
The bandwidth of the input analog stage will be 60 MHz at least (for gain less than 2).
In the actual application the signal will be AC coupled, the input amplifier is set with $A_V = 1$ and the swing of the input signal is reduced to 1 Vpp, to meet AD converter input characteristics.

The AD converter will be a 10 bit device with 5 stages of internal pipeline.
The input signal, coming from the front panel SMB male connector, will arrive to the input stage via coaxial cable RG178 like and locally terminated with cable characteristic impedance of 50 Ω. In this way we minimize all possible effects due to crosstalk and inducted noise.
The converted data (10 bit and over-under range indicator) will then stored in a synchronous FIFO, 256 word deep.
In parallel with data storing the corresponding time tag will be stored in analogous structure: the 16 bit word will be written in 2 times, 8 LSB first and 8 MSB last, in a 8 bit component.
The control logic will be implemented in a programmable component for flexibility and compactness. It starts when the discriminator signal fires, wait for a programmable number of cycles (between 2 and 8) before activate the control signal for writing and then write another programmable number of data into the FIFO (between 2 and 16).
The time tag data is written in parallel with the first two data sample.
Two additional control signal are foreseen: ENABLE and TEST. They are useful to enable the single channel and to start the write sequence in a software controlled way. These signals are independent for each channel.
The actual implementation of the control logic needs a clock cycle to restart itself: this implies a dead time (or better a "blind" time) of 25 nsec after a complete event.
The control signals for reading the two FIFOs are independent. The read and write clocks are independent again and they are distributed in a star topology using impedance controlled traces with $Z_0$ in the range of 50 Ω and 100 Ω.

Each channel will be implemented on a separate board and then connected to the main board using multicontacts connectors. From now on we speak in the term of piggy-back card.
This solution has a little disadvantage in term of production costs, but several advantages: development and prototype testing simpler, production test simpler, spare parts maintenance simpler.
It is also possible implement the piggy-back structure to minimize inducted noise and RF components, to keep as low as possible the crosstalk between channels (on board and in different neighboring cards).
The AD channel is also interchangable: if some application needs different characteristics, it is possible to set up a board with different channels, but keeping the same structure. For this reason the complete 16 bit data bus will be connected to the channel also if only 11 bit are used in this configuration.
The channels piggy-back will be mounted on the mother board using multicontacts high density connectors: the clearance between the two component planes will be of about 10 - 12 mm: it is then possible to mount the components on the two boards one against the other, keeping the last upper plane of the "sandwich" for EMC shield purpose.
The actual foreseen dimensions for the channel piggy-back are about 50 mm x 70 mm.

In Tab. 1 the main components of the AD channel are listed.

For passive components format 0805 will be used as much as possible.

| *COMPONENT* | *MANUFACTURER* | *PART-NUMBER* | *PRODUCTION COST ( +VAT)* |
|---|---|---|---|
| **AD converter 10 bit 40 MHz** | Analog Devices | AD9050BRS | 22500 |
| **Input amplifier** | Analog Devices | AD830JR | 5000 |
| **Synchronous FIFO 256 x 9 (n. 3)** | IDT | IDT72201-15J | 23500 |
| **Control logic** | AMD | MACH211-10J | 17200 |
| **Input protection device (n. 2)** | National | BAV99 | 250 |
| **60 pin male connector** | AMP | 173280-4 | 4630 |
| | AMP | 174216-4 | 5800 |
| | AMP | 1-103916-4 | 4900 |
| | HIROSE | FX4C3-60P-1.27DSAL | 2450 |
| **60 pin female connector** | AMP | 173279-4 | 5180 |
| | AMP | 174215-4 | 8000 |
| | AMP | 104078-6 | 6650 |
| | HIROSE | FX4C2-60S-1.27DSAL | 2800 |
| **SMB panel connector** | SUHNER | 22 SMB-50-0-3 | 4150 |
| | RADIALL | R 114 554 | 2150 |
| | CPE | 22.004.004.005 | 2200 |
| **PCB terminal for RG178** | SUHNER | 86 MCX-0-1-1 | 9380 |
| | CPE | 80.5420.017 | 2800 |

**Tab. 1**

On the main board the channels are grouped by 4, buffering the buses (output: converted data and time, input: time tag).
For every group of 4 channels an ECL to TTL converter (MC10H125) is used to generate TTL discriminator signals.
ECL input signals can be terminated or not ($Z_0$ = 120 Ω), independently for each channel, placing or removing a jumper.

The AD converters need a precise and stable reference voltage to work properly.
We decide to generate an unique reference and then distribute it to all channels, instead of having 32 independent signals. This is done to keep maximum control of the whole circuit.
We use a precision 2.5V reference generator and then an analog unitary gain buffer for each channel to drive the corresponding line along the board.
On the channel piggy back there is a low-pass filter (simple RC) to minimize the possible inducted noise on the PCB trace.
All this circuitry is designed keeping in mind that our LSB is 1mV, so precise and stable components was chosen.

## Main Control

In Fig. 2 a block diagram of the whole board is presented.
The DSP (Texas Instruments TMS320C50) will run at the same frequency of the channels (AD converters and FIFOs), nominal clock frequency will be 40 MHz: the board will be completely synchronous (except VME bus and discriminators input signals, of course).
Since DSP has a 16 bit bus architecture, all interfaces to it have been made with its bus size.
The DSP will be interfaced to the channels for readout data using programmable logic components with $t_{pd} \leq 7$ nsec in order to implement zero wait states accesses. To do this way we need also to adjust the delay of the read clock for the channel FIFOs (see later about clock distribution).
This interface is done in parallel using several small components instead of a big one, because “smaller is faster”.

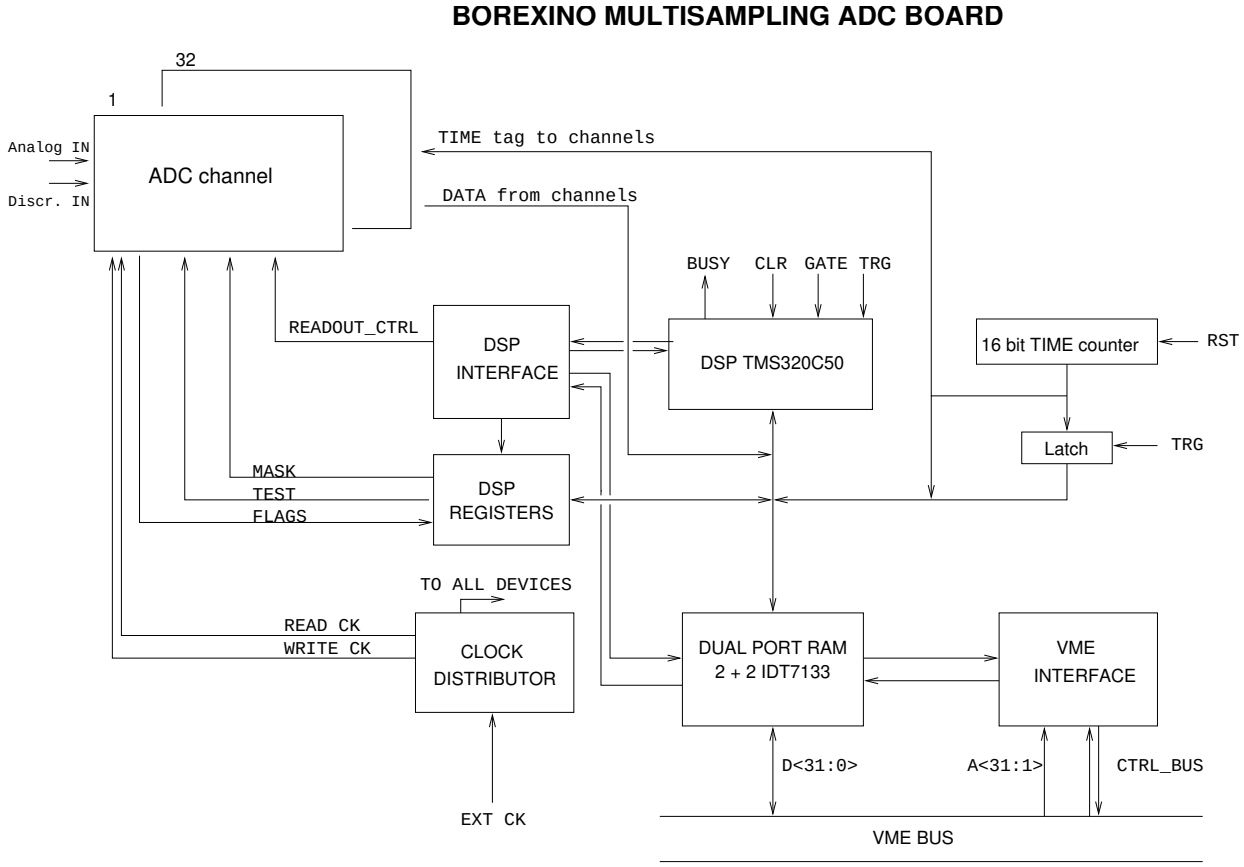


**Fig. 2: ADC board block diagram**

The DSP is then interfaced with several registers used to read the channels status (EMPTY and FULL), to set the working conditions (ENABLE channels, set how many clock cycle wait before write data and how many data has to be written), to read the running time tag counter (free running and latched after a trigger).
The registers and the time counter (incremented at 20 MHz rate) are implemented in FPGA components with medium size and speed.
The time counter is also writable, in particular for debugging purposes.

The access to the dual port RAM buffer (VME accessible, from the other side) is implemented keeping into account the BUSY feature of the DPRAM components: when an access is decoded to a busy location, the READY signal to the DSP is negated, so the DSP itself adds wait states until the addressed location becomes accessible again.

A FLASH EPROM is foreseen to bootstrap the DSP. It is possible to implement software routines for downloading of new software applications through VME bus via DPRAM.

The DSP will be interrupted in these situations in order of priority:

- External Trigger: coming from the external control connector (highest priority)
- VME int2: generated by VME interface when a write access to 0x8004 is detected
- VME int3: generated by VME interface when a write access to 0x8008 is detected
- External Reset: coming from the external control connector (lowest priority)

The DSP NMI signal is not used on this unit.
The DSP BIO input signal is connected to the GATE signal coming from the external control connector: it can be used to indicates the time window for storing data.
The DSP XF output signal is connected to the BUSY signal to the external control connector: it can be used to signal the busy condition of the board to an external logic (trigger logic, for example) and eventually generate an interrupt to the VME crate controller CPU (see later on VME interface).
The functions of these signal is completely software controlled.

In Tab. 2 are listed the main components for the DSP section, together with the voltage reference generator circuit and the ECL to TTL interfaces.

| *COMPONENT* | *MANUFACTURER* | *PART-NUMBER* | *PRODUCTION COST (+VAT)* |
|---|---|---|---|
| **DSP** | Texas Instruments | TMS320C50PQ | 57500 |
| **Flash EPROM 64K x 8** | SGS | 28F512 | 6500 |
| **Channel decoders (n. 6)** | AMD | MACH211-7JC | 22000 |
| **DPRAM decoder** | AMD | MACH211-10JC | 17200 |
| **Registers (n. 3)** | XILINX | XC3142-4PQ100C | 32000 |
| **Data Buffers (n. 16 + 2)** | IDT | IDT74FCT16245CTPV | 4370 |
| **TTL to ECL** | Motorola | MC10H124FN | 6450 |
| **ECL to TTL (n. 9)** | Motorola | MC10H125FN | 6450 |
| **34 pin connector (n. 2)** | HIROSE | HIF3FC-34PA-2.54DS | 1500 |
| **Voltage reference** | Burr Brown | REF1004C-2.5 | 2400 |
| **Analog Buffers (n. 8)** | Burr Brown | OPA404KU | 17700 |

**Tab. 2**

The signals on the external control connector (Trigger, Gate, Reset, Clock, Busy) are differential ECL.
The receiver will be a MC10H125 and the transmitter a MC10H124.
The ECL input signals can be terminated or not ($Z_0$ = 120 Ω), independently, placing or removing a jumper.
The ECL output signals are connected to $V_{EE}$ with 390 Ω resistors.

## Clock Distribution

All devices on the board are synchronous, so a single clock must be distributed everywhere: channels, DSP, control logic, etc.
The AD channels will have independent write and read clocks, to implement the possibility to use 4 channels in parallel and use them as 160 MHz digitizer (connecting the same inputs, analog and discriminated, to the 4 channels, via an external fanout).
For the channels clock distribution we decide to implement a star topology, keeping independent the clock lines for every channel: this is done for load balancing and to reduce the skew time between channels in the order of 1 nsec.
The frequency of this clock is 40 MHz, so particular care must be taken into account in the design of clock distribution circuits, both from the electrical and physical point of view.

We decide to have three sources of input clock:

- differential ECL signal, connected to the external control connector
- single ended TTL signal, connected to a front panel coaxial connector
- on board crystal oscillator

The selection between the three sources is done placing a jumper.
Differential ECL clock can be terminated or not ($Z_0$ = 120 Ω), placing or removing a jumper.
Single ended TTL clock is terminated with coaxial cable characteristic impedance ($Z_0$ = 50 Ω).

The selected clock is then regenerated using a dedicated PLL circuit with 8 output lines.
Some of these lines feed the clock input of DSP and control logic, two of them are used to generate the read and write clock for the AD channels.
In Fig. 3 the block diagram of the clock distribution circuit is presented.

**CLOCK DISTRIBUTION CIRCUIT**

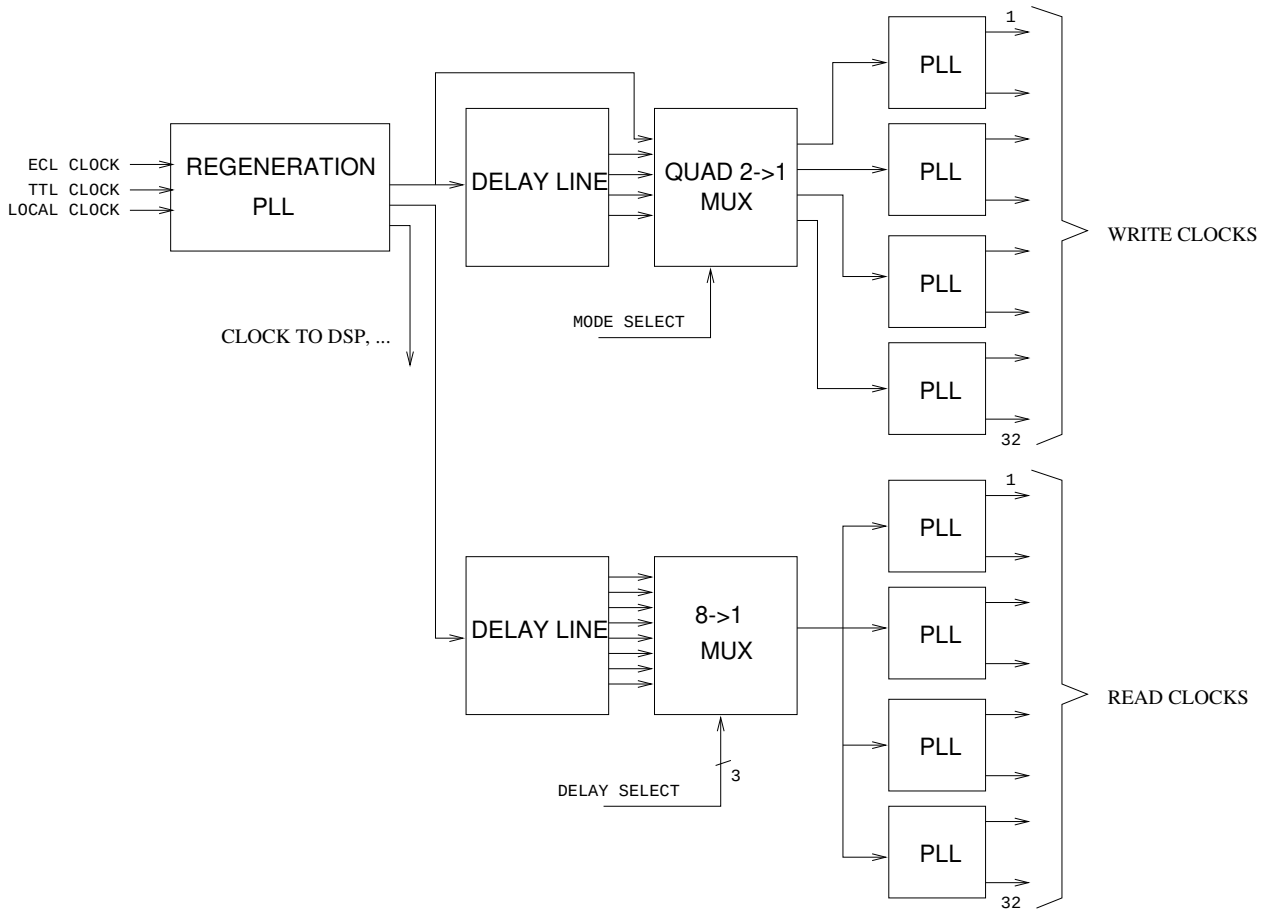


**Fig. 3: clock distribution block diagram**

To generate the write clock to the AD channels with the possibility of implement the 160 MHz digitizer, we use a tapped delay line, extracting 4 signals, with relative delay of about 6 nsec (about a quarter of 25 nsec period).

A multiplexer is then used to select the operating mode of the board: 32 channels @ 40 MHz or 8 channels @ 160 MHz.
The 4 clock signals are then connected to corresponding PLL circuits (same as the first one) and the outputs of these components are connected to the write clock lines of the channels.

The generation of the read clock to the AD channels is very similar to the preceding.
In this circuit we must implement the feature to control the read clock phase respect to the DSP clock, in order to have zero wait states accesses to the channel FIFOs.
We use a tapped delay line and a multiplexer to select the exact delay for the final clock distribution.

We need to measure the time relationship between the rising edge of the read clock and the read enable FIFO signal to ensure that set-up and hold times are guaranteed: this must be done at the board power up and during diagnostic operations.
The control firmware must do this measurement and then set the correct value on the multiplexer input to generate an affordable clock for reading the channels FIFOs.
It is not possible guarantee by design to satisfy the FIFO timing requirements because the delay in the generation of DSP external bus control signals respect to the input clock is defined with about 20 nsec spread. This is comparable with our clock period and hence if we want to run the DSP interface to the channels at zero wait states we need to tune the read clock in respect to the DSP generated signals.

We foresee two methods to carry out this measurement: completely software or with some help from hardware.

A possible software method is the following.
1. Disable all channels and set them to write a known number of data word (i.e. four).
2. Set the delay to read clock to the minimum possible.
3. Reset all channels to start with a known condition (no words written into the FIFOs).
4. Generate some test pulses to all channels (i.e. two).
5. Read data from all channels FIFOs, checking the Empty Flag every read operation: if a FIFO becomes empty before or after the expected number of reads a timing violation has occurred. In this case we can set the delay two steps before or after the actual and that's all.
6. Increase one step of delay and repeat the test generation/reading sequence (steps 3, 4 and 5).

A possible hardware circuitry to do this kind of measurement is based on a pulse width discriminator component (PWD).
The PWD pass on its output the input signal if its width is greater than a minimum (component characteristic) otherwise the output will be zero.
Using a set-reset flip flop to create a pulse with width equal to the set-up time, feeding the PWD and connecting the PWD output to a retriggerable one shot it is possible to see when the set-up time is satisfied or not.
An analogous circuit can be set for hold time measurement.
The software routine should issue some read pulses and change the delay in read clock generation circuit until both monostables output are high.

The overall precision of the components used in the board limit the number of channels that can be grouped together to form a N times digitizer.
To implement the clock distribution circuit we cannot use passive delay lines because the minimum pulse width they guarantee to pass through is greater than our clock pulse. We need to use fast silicon delay lines which guarantee to pass clock signals over 100 MHz.
These components have intrinsic precision that are not very good (relative time between successive taps in $\pm$ 1 nsec range) so we cannot group more than 4 channel together.
Another limit for channel grouping is the aperture delay of the AD converter ( < 3 nsec).

The channels clock traces (both read and write) have controlled impedance ($Z_0$ = 50 $\Omega$) and terminated on the piggy backs, near the loads.

In Tab. 3 is presented the component list of clock distribution section.

| *COMPONENT* | *MANUFACTURER* | *PART-NUMBER* | *PRODUCTION COST (+VAT)* |
|---|---|---|---|
| **PLL (n. 9)** | Cypress | CY7B9920-5SC | 19000 |
| **Delay line (n. 2)** | Data Delay Devices | 3D7110S-3 | 24700 |
| **Quad 2 input mux** | IDT | IDT74FCT157CTSO | 3850 |
| **8 input mux** | IDT | IDT74FCT151CTSO | 3960 |
| **PWD (n. 2)** | Data Delay Devices | PWD-21-5 | 20880 |

**Tab. 3**

## VME Interface and Memory Map

The VME interface is composed by:

- board base address decoder
- bidirectional data bus buffers
- 16 Kbytes (4 Klongword) Dual Port Ram
- address decoder and control logic

The board base address is decoded comparing the Jaux geographical address signals against the five address bits A20..A16. The address bits A23..A21 must be set to zero.
Additional flexibility is added using an 8 bit cabled address (which will be set placing 8 jumpers) and comparing it with address bus bits A31..A24. Each jumper placed set the corresponding bit to 1.
The bidirectional data buffers are used to decouple the internal bus to the backplane bus. They are implemented using 16 bit components for compactness.
The Dual Port Ram is organized in 32 bit longword. Four chip are foreseen (2 Kword x 16 bit each), but if only 8 Kbytes are needed, the upper 8 Kbytes cannot be mounted without any implication.
The address decoding and control logic is implemented in a FPGA with medium size and speed, for flexibility and compactness.
It decodes the lowest 15 bit of the address bus, recognizes various VME cycles and generate the corresponding control signals for handling on board devices.
This card is a VME slave device: no master capabilities are implemented. In VME terms it can be defined as A32, D32, BLT.
Only extended (32 bit) addressing is supported. Standard and short addressing (24 and 16 bit, respectively) are not supported. The board will respond to privileged and non privileged data and program address modifiers (AM codes: `09, 0A, 0B, 0D, 0E, 0F`).
Only longword (32 bit) aligned data transfers are supported. Unaligned data transfers, word transfers and byte transfers are not supported.
Single VME cycles and pipelined single VME cycles are supported, block transfers are supported as well, for Dual Port Ram accesses.
The board could be also an interrupter D08(O), ROAK, but it is not implemented in the first release.
Interrupts are controlled through a status/control register in which the user can set the IRQ level, the STATUS/ID to be placed on VME in the IACK cycles and an interrupt flag.
Interrupts are generated, if enabled on the specified level, on the rising edge of the DSP XF signal (unit BUSY condition). The status of the XF line is available as interrupt flag.
In Tab. 4 a list of the VME interface component is reported.

| *COMPONENT* | *MANUFACTURER* | *PART-NUMBER* | *PRODUCTION COST (+VAT)* |
|---|---|---|---|
| **Data Buffers (n. 2)** | IDT | IDT74FCT16245CTPV | 3700 |
| **Board decoders (n. 2)** | IDT | IDT74FCT521TSO | 3200 |
| **DPRAM (n. 4)** | IDT | IDT7133SA35J | 30600 |
| **Control Logic** | XILINX | XC3142-4PQ100C | 32000 |
| **Connectors 96 pin (n. 2)** | ERNI | 533-402 | 3800 |
| **Connector 30 pin** | ERNI | 424-189 | 2450 |

**Tab. 4**

The on board devices are mapped as follow:

| | |
|---|---|
| DPRAM | 0x0000 - 0x3FFF |
| Status/Control Register | 0x8000 |
| VME int2 to DSP | 0x8004 |
| VME int3 to DSP | 0x8008 |
| Software Reset | 0x800C |

Space between 0x4000 and 0x7FFF and between 0x8010 and 0xFFFF is unused and accesses here generate bus errors.

The format of the Status/Control register (used for IRQ handling) is the following:

| | |
|---|---|
| STATUS/ID | bit 7..0 |
| IRQ level | bit 10..8 |
| IRQ flag | bit 11 |

Bits from 10 to 0 are both writable and readable; bit 11 is read only and is the status of DSP XF line. Interrupts can be generated only on rising edge of this signal, if enabled. In this release interrupts on VME bus are not generated.
The STATUS/ID fields is the value placed on the VME bus lines D07..D00 in IACK cycles.
The IRQ level field value sets the level of generated interrupts: if it is reset to 000 interrupts are disabled (default)

Any write access to 0x8004 cause the generation of INT2 to DSP.
Any write access to 0x8008 cause the generation of INT3 to DSP.
Any write access to 0x800C cause the generation of a RESET signal to the whole board (same effect as power up reset).

## DSP Memory Map

The Texas Instruments DSP TMS320C50 has three independent addressing spaces: Program space, Data space and I/O space.
To decode various on board devices, seen by the DSP we use only data space and I/O space. Program space is used only to boot the DSP using the external flash EPROM.

The Dual Port Ram and the channels FIFOs are mapped in data space, while the registers are mapped in I/O space.

*DPRAM (in Data Space)*

| | |
|---|---|
| 0x4000 - 0x47FF | (Corresponding VME address 0x0000 - 0x1FFF bits 0..15) |
| 0x5000 - 0x57FF | (Corresponding VME address 0x0000 - 0x1FFF bits 16..31) |
| 0x6000 - 0x67FF | (Corresponding VME address 0x2000 - 0x3FFF bits 0..15) |
| 0x7000 - 0x77FF | (Corresponding VME address 0x2000 - 0x3FFF bits 16..31) |

*CHANNELS (in Data Space)*

| | |
|---|---|
| 0x8000 - 0x801F | Data FIFO for channels 0..31 |
| 0x8020 - 0x803F | Time tag FIFO for channels 0.. 31 |

To keep the decode logic as simple as possible the channels FIFOs are remapped N times between 0x8000 and 0x8FFF.
To read a complete 16 bit Time tag, two accesses are needed: the first reads the 0..7 bits and the second reads the remaining 8..15 bits.
In both cases (Data FIFO & Time tag FIFO) not used high bits are set to 0.

*REGISTERS (in I/O Space)*

| | | |
|---|---|---|
| 0x0050 | Empty flags register for channels 0..15 | read only |
| 0x0051 | Empty flags register for channels 16..31 | read only |
| 0x0052 | Full flags register for channels 0..15 | read only |
| 0x0053 | Full flags register for channels 16..31 | read only |
| 0x0054 | Enable register for channels 0..15 | read write |
| 0x0055 | Enable register for channels 16..31 | read write |
| 0x0056 | Unlatched time register | read write |
| 0x0057 | Latched time register | read write |
| 0x0058 | Program register | read write |
| 0x0059 | Test register for channels 0..15 | write only |
| 0x005A | Test register for channels 16..31 | write only |
| 0x005B | Reset all channels | write only |

To keep the decode logic as simple as possible the registers are remapped N times between 0x0000 and 0x0FFF every 256 words boundary.

Program register (0x0058) bit assignment

| | |
|---|---|
| PROG[2..0] | N. of wait cycles before write data samples (1..7 correspond to 2..8 wait cycles) |
| PROG[6..3] | N. of data samples to write (1..15 correspond to 2..16 samples) |
| PROG[7] | Connected to the channel piggy back, but not used in this configuration |
| PROG[8] | Mode of operation (0 = 32 channels @40Mhz, 1 = 8 channels @160Mhz) |
| PROG[11..9] | Delay for read clock generation (3 nsec/step) |
| PROG[13..12] | Unused (connected as inputs) |
| PROG[15..14] | Setup-hold time check for channel FIFOs read enable signals |

Test pulse generation

setting bits in registers 0x0059 and 0x005A activate the generation of a 50 nsec test pulse for the corresponding channel.

## Firmware Control Operations

The firmware running on the DSP should carry out these operations:

- Set up working condition for the board
- Do board diagnosis
- Build up histograms and do other statistical measurements
- Read-out data from channel and store them in DPRAM after reformatting on a trigger basis algorithm

The routines used to carry out the first point use configuration data stored in the dedicated DPRAM bank. These routines are triggered upon receiving INT2 or INT3 (this is not chosen yet).
First the VME crate controller writes configuration data or particular commands in dedicated DPRAM locations and then issue an INT2 or INT3 writing to 0x8004 or 0x8008, respectively.
The DSP is interrupted, performs the requested operations, eventually sets data to read back and finally notifies the end of the requested operation setting a flag.

Board diagnosis routines are activated at power up, on crate controller requests (via INT2, INT3 mechanism) and could also run while data taking is in progress to control that all on board devices are working properly.
Diagnosis routine must also measure the set-up and hold time for read clock generation and set the correct delay.

Histograms are build up upon crate controller requests in a software controlled way (details are not defined yet), and data are stored in the dedicated DPRAM bank.

Data taking is the main purpose of this board and must be carried out in the most efficient way in order to have the maximum possible input frequency, without data losses and without introducing significant dead time in the trigger generation.

The main program flow during data taking is to keep the channels FIFOs almost empty.
The DSP will cycles through all the enabled channels, check for FIFO empty flags and if not empty read both time tag and the programmed number of converted data.
The time tag is compared with the current time (reading the time counter): if the time difference is greater than the trigger latency the data are discarded, otherwise they are stored in a local circular buffer.
At this stage the data can be used for on board measurements and histogramming.
For each channel a circular buffer is allocated in order to have a second level FIFO before generating the output data record on trigger requests.
The dimension of these buffers should be as greater as possible: it can be determined statically, keeping into account all 32 channels and the maximum number of data words (16) or dinamically (each run) for the enabled channels and the effective requested data words.
In the first case we need (16+1) x 32 words (16 bit) for each event: using the internal DSP RAM we can house about 10 levels of buffering for each channel.
When the circular buffers overrun oldest data are lost.
Considering maximum input frequency of 1 MHz for each channel, we can store at least 10 μsec before buffers overrun.
Trigger latency is foreseen in the order of 1 μsec.

Considering Borexino conditions (4 samples for each input signal) we can estimate a mean time of about 1 μsec to execute the routine for checking each channel and eventually copy the data in the corresponding circular buffer.
The mean channel input frequency is then in the order of 30 Khz. Peak frequency is handled by the input FIFO and can be estimated to be up to 10 times greater than mean value, without data losses.

In case of **trigger** the DSP will be interrupted at highest priority level (maskable).
The interrupt routine should be kept as simple as possible to have the possibility so serve a high trigger rate (1 Khz mean, at least).
In the minimal configuration it could only read the trigger time, store it in a trigger queue and increment an non served trigger counter.
The main program will check the non served trigger counter and if it is greater than zero call the reformatting routine to generate output data in DPRAM.
The reformatting routine must scan the channels circular buffer in time reverse order, find the first data belonging to the trigger in process, discarding the others, and copy the data into the DPRAM.
It must also check for acquisition time window (using the BIO line or software settings) and at the end increments internal counters and set some variables to keep track of the data.
The software must also keep track of event numbering, incrementing a 16 bit counter every received trigger.
The external reset signal interrupts the DSP: the data acquisition process terminates and the board goes in an idle state.

The DSP can signal particular conditions setting the XF flag.
This signal can be used to generate an interrupt on VME bus and sent directly to some external control logic.
These operation can be independently enabled or not via jumpers (interrupts must be also enabled by software).

## Data Formats

We try to define the possible data formats for the board. This is very preliminary and must be defined when the real DSP firmware will be developed.

First of all we decide to use three separate bank in the DPRAM. The contents is the following:
1. Configuration data and flags
2. Histogramming and statistical data
3. Readout data

The first bank will contains configuration data such as channels enable pattern, test pulse pattern, trigger latency time, acquisition time window, mode of operation, number of wait cycles and data samples to store, flags, pointers, counters, ...
In addition we define a particular location used to issue commands: diagnostic, test pulse generation, set run mode, reset counters, ...
At this moment we can foresee to reserve the first 16 words (32 bit) for this purpose.

The histogramming and statistical data can be stored in 1024 words (32 bit), because we are using 10 bit components. We could save some space using 16 bit words: this depends on the kind of measurements we'll need to carry out and will be defined in the final software development stage.

The remaining of the DPRAM has to be used to store the converted data. The available space will be less than 7 Kwords (32 bit), but is sufficient for any kind of acquisition process.
The format of the data can be defined using the following BNF syntax:

```
<data>                    :== { <event record> }0
<event record>            :== <trigger number> <total number of words>
                              { <channel record> }0
                              <trailer zero>
<channel record>          :== <channel number> <relative time to trigger>
                              { <word pair> }1
<word pair>               :== (<data word> <data word>) | (<data word> <zero>)
<data word>               :== converted data (16 bit)
<zero>                    :== 0000_0000_0000_0000 (16 bit)
<channel number>          :== unsigned 16 bit number
<relative time to trigger>:== signed 16 bit number
<trigger number>          :== trigger counter (16 bit)
```

```
<total number of words>    :== unsigned 16 bit number
<trailer zero>             :== 0000_0000_0000_0000__0000_0000_0000_0000 (32 bit)
```

All fields are aligned on a 32 bit boundary to simplify and speed up the transfer on the VME bus.
The *total number of words* field is the number of 32 bit words which form a complete events, including itself and the trailer. So it is always grater than 1.
If an event contains no data words the trigger number and the trailer will be always present in the *event record* to ensure consistency between all boards.

The data are organized in a circular buffer fashion an hence two pointers are needed to keep track of the data buffer.
Write pointer is set by the DSP and read pointer is set by the VME crate controller.
The DSP must check for pointers overtaking and eventually set a flag and/or interrupt the VME controller via the XF signal (this operation can also inform the trigger system).
The VME crate controller must check for pointers overtaking to see when all data in the DPRAM buffer have been read.
The VME CPU can poll the two pointers and decide to perform a data transfer only when there is a number of data so a block transfer can be run in the most efficient way.

## Acknowledgement

We would like to mention here the persons that helped us in this job.
First of all our collaborators from EES SrL: without their help we were not able to have so many technical details and the foreseen time schedule could not be satisfied.
Prof. S. Vitale and all of Borexino people from Genova gave us some idea to keep the design as general as possible.

## References

- Borexino technical proposal
- The VMEbus Specification
- TMS320C5x User's Guide
- Various handbook and manuals from components manufacturers (AMD, IDT, Cypress, Analog Devices, Xilinx, Data Delay Devices, ...)