32 channels TDC VME board User Manual*

E. Gennari, M. Passaseo, E. Petrolo, R. Vari, S. Veneziano
INFN-Rome, Italy

January 29, 1998

The 32 channels TDC VME board is a 6U VME module based on the KLOE TDC integrated circuit. So, it inherits all the ASIC's specifications adding the on-board facility of setting the number of chip's buffers in use, a BUSY signal is generated when the requested number of buffers is full. We suggest then to read the chip related documentation. Since block transfer is not implemented on this board, the "Event request readout mode" is not available. The board is a A32-D32 slave module.

1 A short description of TDC functionality

The TDC integrated circuit is developed as a full-custom device in 0.5 μm CMOS technology. All the chip inputs accept TTL signals except the reference clock that is positive ECL. Translators are used to interface the board ECL logic with TDC chip TTL logic.

It's working frequency is a function of an external reference clock that must be supplied to the circuit; on this board the reference clock is 160 MHz, giving a working frequency of 960 MHz for the graycounter, and 1.04 ns LSB. The circuit is capable of detecting rising/falling edges, with a double edge resolution of 8 ns.

The hits are recorded as 16 bit words and every registered hit is stored in the chip for a programmable time interval (RANG register). Afterwards they are discarded, if not confirmed by a stop signal. The number of hits recorded per channel is programmable (MAINCONTROL register). The chip has four

*http://sunset.roma1.infn.it/tdc

event-buffers, which are used only if at least one hit is present in one of the 32 channels for a given trigger.

Dedicated prompt trigger output lines inform the readout of the presence in the chip of the data to be read. The empty channels are automatically zero-suppressed during the readout phase and for each registered hit the chip gives is absolute time value with respect to the stop signal and its channel number. Additional chip functions include an event counter and masking of the individual channels.

1.1 Data readout

During the data taking the chip can be read out in two different modes: "sequential" and "upon event request"; we describe here the "sequential" one, since it is the only one implemented on this board.

As soon as one buffer is filled and the empty buffer flag is unset (EMPTY), the chip presents, on the 16 LSBs of the data bus, the event number corresponding to that particular buffer, on that first read command. Then the data for that event are output in ascending order of channel number; in conjunction with the end of the data a dedicated line (EOB) is set. At this point the event buffer is empty.

This readout sequence can be repeated for other buffers until all the data are read, in the readout sequence the oldest event is output first. So the TDC can be emptied monitoring the EOB and EMPTY lines.

Data are formatted with each buffer beginning with the event number and ending with an end-of-block signal. In order to read directly the time differences between common START and hit or between hits and common STOP, a subtractor circuit is implemented in the chip, giving always positive time differencies in binary format.

1.2 Operating modes

1.2.1 Common Stop

At power up or after a reset command, the read and write lines point to buffer 0, and the content of a continuosly running clock are written into the first set of FIFO's every time a hit comes from the chamber.

A check is done continuosly to see if the counter has stepped the full range (set by an internal register, maximum 32 μs), respect to the oldest hit. The comparison is done disregarding the 5 LSBs. If this is the case a local read FIFO command is issued to the currently written buffer and the oldest hit is discarded.

If the number of hits recorded is bigger than the maximum number (stored in a internal register), an overflow bit is assigned to the channel tagging, during the readout phase, all the hits belonging to this buffer.

In such an arrangement each hit information is 18 bits wide; the 16 LSBs have the time information, bit 16 has the signal polarity (rising or falling edge) and the MS bit is the overflow bit (see par. 4.1). This operation continues until a STOP signal comes into the TDC. The leading edge of the signal is latched. When the stop signal is active no more hits are written into the FIFO. The current FIFO is then freezed by changing the event buffer write lines and the next set of FIFO's are enabled to receive hits.

1.2.2 Common start.

The common START mode operation is similar to the one already described except for allowing hits to be recorded only when the common signal is active; this is achieved by controlling the buffer selection externally.

2 Hardware description

All front panel input signals are differential ECL and are asserted when the pin labeled "+" is ECL high and the pin labeled "-" is ECL low. Translators are used to interface the board ECL logic with TDC chip TTL logic. Only 16 of the 17 pairs of pins of the two Scotchflex connectors for the channels input are used, the topmost pairs are not connected. Channel 0 and channel 16 are the lowermost pins of the connectors. A 160 MHz quartz clock (PECL logic) is mounted on the board to give the chip the reference clock.

The BUSY output is a NIM level that is set when all selected buffers contain data. The BUSY signal must be used to inhibit the START/STOP signal sent to the TDC. The number of buffers in use can be selected through jumpers setting (see par. 2.2)

VME_DECODE block (a PAL ALTERA chip) uses VME backplane signals and TDC chip output signals to manage the board logic and to provide to the TDC chip Enable, Data Strobe and Read/Write signals. A 10 MHz clock is used by the PAL for the signals timing.

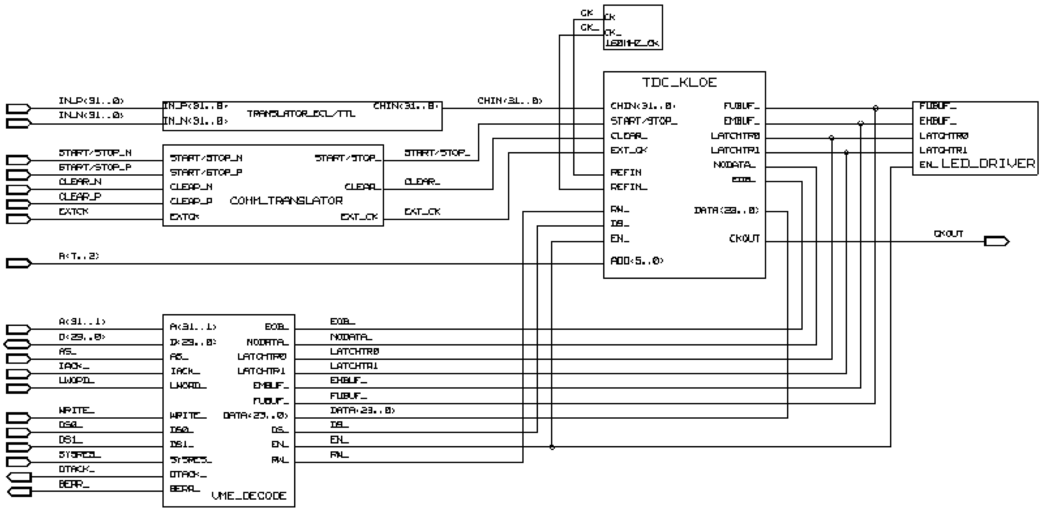

Figure 1: TDC board blocks scheme

Full buffer (FULL), Empty buffer(EMPTY) and Latch lines (L1,L0 - buffers occupancy) of the TDC chip are connected to four LED mounted on the board front panel (light on = signal active). The fifth LED shows VME access.

The FULL LED is on when all the four chip buffers are occupied. The EMPTY LED is on when no buffer is occupied. When this LED is on the chip should not be read since the data would be meaningless. The L1, L0 LED indicate the number of occupied buffers. It does not indicate buffers are occupied.

One buffer is occupied when it contains at least one hit confirmed by a STOP.

L1	L0	buff. occ.
OFF	OFF	check FUBUF and EMBUF
OFF	ON	1
ON	OFF	2
ON	ON	3

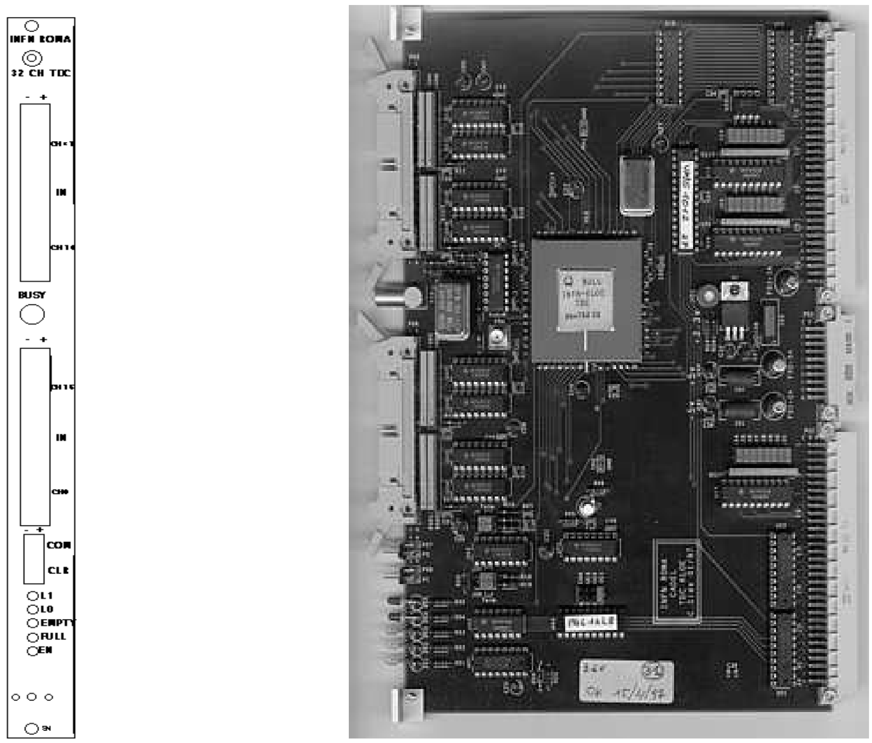

Figure 2: Front panel and TDC board

Signal	*Type*	*Location*	*Description*
$CHIN[31:0]$	IN	Front Panel	Channels
COM	IN	Front Panel	Start/Stop
CLR	IN	Front Panel	Clear
$BUSY$	OUT	Front Panel	BUSY
$D[29]$	OUT	VME Connector	Full Buffer Signal
$D[28]$	OUT	VME Connector	Empty Buffer Signal
$D[27:26]$	OUT	VME Connector	Latch
$D[25]$	OUT	VME Connector	No Data
$D[24]$	OUT	VME Connector	End Of Block
$D[23]$	OUT	VME Connector	Event Number Flag
$D[22:0]$	INOUT	VME Connector	DATAIO[22:0]
$A[31:8]$	IN	VME Connector	VME Board Address
$A[7:2]$	IN	VME Connector	ADD[5:0]
$A[1]$	IN	VME Connector	VME longword sel
$CKOUT$	OUT	On Board	Clock Output

Table 1: I/O signals description

2.1 Signals timing

The channels (CHIN) signals must be differential ECL signals with a duration of at least 8 ns. Two active edges must be separated by at least 8 ns.

The CLEAR (CLR) signal is differential ECL signals, its duration must be at least 32 ns.

The STOP (COM) signal is differential ECL signals, its duration must be at least 32 ns. The leading edge of the STOP signal is latched. In Common Start Mode, the leading edge marks the beginning of the time interval in which hits are recorded. In Common Stop Mode, the leading edge marks the end of the time interval in which hits are recorded. Two front panel inputs are available for daisy chaining several modules together.(see sec 2.2)

2.2 Jumpers setting

Three sets[1] of jumpers are used to set the VME base address[2] of the board. (see table 2)

	Jumpers	*VME address line*
A	J15-J08	A15-A08
B	J23-J16	A23-A16
C	J31-J24	A31-A24

Table 2: Address jumpers setting

The COM and CLR inputs must be terminated if not used in daisy-chain.(see table 3)

The user can decide to use up to 4 buffers(events) of the TDC chip. Only one jumper can be set. (see table 4)

3 Internal registers and commands

All TDC chip internal registers can be accessed through read VME cycles, most of them can be set with a write VME cycle, depending on the status of the flag TESTMODE in the MAINCONTROL Register.

[1]See fig 4.3 for A-F flags.

[2]Active line = "1" = no jumper.

	Jumpers	*function*
D	J5,J7	Common $START/STOP$ input terminators active termination = jumpers on the right side(marked term) no termination(for daisy chain) = jumpers on the left side
E	J32,J34	CLEAR input terminators active termination = jumpers on the right side(marked term) no termination(for daisy chain) = jumpers on the left side

Table 3: COM and CLR terminations jumpers

	Jumpers	*BUSY set after n buffers full*
F	J1	=1
	J2	=2
	J3	=3
	J4	=4

Table 4: BUSY line setup

Commands are issued with a read/write VME cycle to the specified address.

Table 5 shows the register address as offset to the VME base address and the access mode.

3.1 Registers

- EMPTY FLAGS (EF)[15:0] : 8 registers. They show the status of the 32 FIFOs (one per channel) for each event buffer (4 event buffers)(1=channel empty - Read only).

- STOP REGISTERS[15:0] : 4 registers. They record the value of the gray-counter when the start/stop signal arrives(Read only).

- REPT[6:0] This register is used to check the status of the buffers read and write pointer, and of the internal stop.

  REPT[6] Internal STOP - REPT[5] Counter Phase - REPT[4] NO-DATA flag - REPT[3:2] write event-buffer pointer - REPT[1:0] read event-buffer pointer

Address	Access	bits	Description	VME OFFSET[7:0]
$FIFO$	R	23	Event buffers	00_{hex}
$EVRQ$	W	16	Event requested	04_{hex}
$GENS$	R(T)W(T)	x	Generate internal STOP	44_{hex}
$MAIN$	RW	16	Main control	08_{hex}
$CREG$	RW(T)	16	Clock control register	48_{hex}
$BECK$	R(T)W(T)	x	Begin DLL command	88_{hex}
$RSCK$	R(T)W(T)	x	Restart DLL command	$C8_{hex}$
$RANG$	RW	6	Counter range	$0C_{hex}$
$EVNU$	RW(T)	16	Event number	10_{hex}
$CONT$	RW(T)	16	TDC counter	14_{hex}
$MASK_i$	RW	16×2	Channel masks	$0\text{x}i6 * 4_{hex}$
$EF0F_i$	R	16×2	Empty flags Ev. buffer 0	$0\text{x}i7 * 4_{hex}$
$EF1F_i$	R	16×2	Empty flags Ev. buffer 1	$0\text{x}i8 * 4_{hex}$
$EF2F_i$	R	16×2	Empty flags Ev. buffer 2	$0\text{x}i9 * 4_{hex}$
$EF3F_i$	R	16×2	Empty flags Ev. buffer 3	$0\text{x}iA * 4_{hex}$
$EVRE_j$	RW(T)	16×4	Event registers	$0\text{x}jB * 4_{hex}$
$STOP_j$	R	16×4	START/STOP registers	$0\text{x}jC * 4_{hex}$
$CLSB$	R(T)W(T)	x	Clear single buffer command	34_{hex}
$REPT$	R	7	Flags + RW buffer pointers	38_{hex}
$CLAL$	RW	x	Clear all buffers command	$3C_{hex}$

T	Read or Write access enabled only if the TEST flag is set
$i = 1..0$	respectively the 16 MS and LS channels
$j = 3..0$	event buffer number

Table 5: Internal registers addresses

- Event counter $EVNU[15:0]$. It contains the number of START/STOP signals received by the chip; it can be used to check the synchronization of different chips.(R/W only in TESTMODE)

- Channel masks, $MASKH[15:0]$, $MASKL[15:0]$. Each mask bit enables single channels.(R/W) (1=channel enabled; 0 = channel disabled)

- Current event number requested by DAQ, $EVRQ[15:0]$ (R/W) (not used on this board).

- Counter range, $RANG[5:0]$. It is used to define the maximum time span to be recorded into an event buffer when in COMMON STOP mode. $RANG[3:0]$: number of effective counter bits (time window): only the hits arriving within (2**(RANG[3:0]+1))*1.04 ns will be kept. (Ex. RANG[3:0]=0xA time window=2048*1.04 ns). Maximum value is 0xE, minimum is 0x6.

  In COMMON START this register must be set to the smallest value greater than the length of the START signal. For example if the START (COM input) signal is 3 μs long, RANG must be set to 3B corresponding to a time window of 4.26 μs

  $RANG[5:4]$: depth of comparison of old event elimination, $00 = disable$, $01 = 8\ ns$, $10 = 16\ ns$, $11 = 32\ ns$. It must be set to 10 or 11 when working in COMMON STOP mode. In COMMON START mode it should be set to 11.(R/W)

- $MAINCONTROL[15:0]$, Working mode register (See paragraph MAINCONTROL register)

- $CREG[15:0]$, clock multiplier register, for monitor and control of the DLL.(R/W in TESTMODE) (See paragraph Clock Setup)

3.1.1 Working mode setup: the MAINCONTROL register

MAINCONTROL register includes the following settings:

NHITS[3:0] Maximum number of hits allowed per event. NHITS[3:0] = 0x0 1 hit allowed NHITS[3:0] = 0xF 16 hits allowed

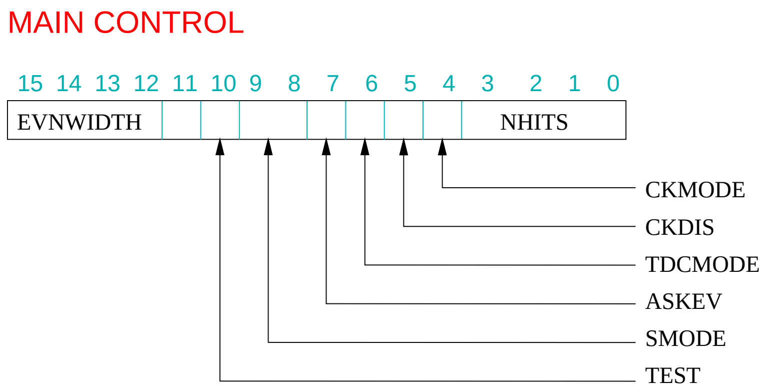

Figure 3: MAINCONTROL register

CKMODE The TDC clock can be internally generated, out of an external reference clock working at moderate frequencies, or externally at the full working frequency.

$CKMODE = 1$ DLL clock
$CKMODE = 0$ external clock.

CKDIS for debug purposes the clock can be disabled by setting the CKDIS flag.

$CKDIS = 1$ clock disabled
$CKDIS = 0$ clock enabled

TDCMODE the TDC can work in common STOP or common START mode.

TDCMODE=1 COMMON STOP
TDCMODE=0 COMMON START

ASKEV The readout of the chip can be done in two different ways, simply by emptying all the event buffers sequentially (oldest event first) or by asking for a particular event number. If the ASKEV flag is set then the chip is set in the event request readout mode. Using this board this bit must always be set to 0.

SMODE[1:0] It is foreseen to record rising edges, falling edges or both, depending on the setting of the signal mode register.

$SMODE = 1$ rising edge only
$SMODE = 2$ falling edge only
$SMODE = 3$ both edges are recorded
$SMODE = 0$ not allowed

EVNWIDTH[3:0] These four bits set the number of significant bits of the Event Registers and of the Event Counter (15 = 16 bit, 0 = 1 bit), used in the event request ($EVRQ$) cycle.

TEST All the registers inside the chip can be read or written. In order not to distrupt the data during normal data taking due to a wrong operation, some commands can be issued only after setting a $TEST$ bit on.

The bits are arranged as follows:

$NHITS$	$maincontrol[3:0]$
$CKMODE$	$maincontrol[4]$
$CKDIS$	$maincontrol[5]$
$TDCMODE$	$maincontrol[6]$
$ASKEV$	$maincontrol[7]$
$SMODE$	$maincontrol[9:8]$
$TEST$	$maincontrol[10]$
$EVNWIDTH$	$maincontrol[15:12]$

3.1.2 Clock Setup

For test purposes a connector, mounted on the board, can be used to access the CKOUT TDC chip pin. The selected clock can be tested with an oscilloscope connected to the output clock of the board.

Flag Name	*Registers Bits*	*Value*	*Action*
CKSOURCEMUL	CREG(15)	0	160 MHz clock input to DLL
		1	External clock input to DLL
CK_FHALF	CREG(10)	0	Full frequency clock
		1	Half frequency DLL output enabled
CKMODE	MAIN(4)	0	Half clock/External clock enabled
		1	DLL clock enabled
CKDIS	MAIN(5)	0	TDC chip internal clock enabled
		1	TDC chip internal clock disabled
CKVALPAD	CREG(11)	0	CKOUT board output disabled
		1	CKOUT board output enabled
CKVALPADN	CREG(12)	0	CKOUT_N board output disabled
		1	CKOUT_N board output enabled
CKPADEXT/ CKPADEXTI	CREG(13/14)	00	CKOUT = "1"
		01	CKOUT = Reference clock
		10	CKOUT = External clock
		11	CKOUT = Internal TDC chip clock

So, to see the 160 MHz, 480 MHz clock, or the External Clock, these are the value of the Clock Register and of the Main Control Register of the TDC chip:

CKOUT	CREG[15:10]	MAIN[5:4]
160 MHz	010110	01
480 MHz	011110	01
external	011100	00
acq mode	000000	01

Table 6: Registers value for clock output setting

In normal acquisition mode CREG[15:10] must be set to 000000 and MAIN[5:4] to 01. CREG[9:0] bits are set by TDC chip.

3.2 Commands

CLSB The event buffer currently addressed by the read pointer is cleared as well as the corresponding STOP register, and event register (EVRE). The read pointer is then incremented.

CLAL is equivalent to an external clear. It clears event registers, event number register, stop registers and all the fifos.

BECK must be used to start the DLL.

RSCK this command has been implemented for clock debugging purposes.

GENS command implemented for chip tests.

4 Data acquisition

4.1 Data Format

The TDC DATA bits correspond to the 22 LSB of the D[31:0] bus. Data from the fifos are read reading at address VMEBASE + 0x00. It is a unique address for all the buffers, you cannot select the buffer to read. The first event output, is the oldest in the 4 buffers.

D[22:18]	channel number
D[17]	overflow flag : 1 = the fifo has overflown
D[16]	edge flag : rising = 0, falling = 1
D[15:0]	TDC data word

D[29-23] reflect the status of the following flags:

D[29]	FUBUF (Chip full if =0)
D[28]	EMBUF (Chip empty if =0)
D[27-26]	LATCHTR (number of buffers full) =0 chip full or empty - check FUBUF and EMBUF =1 1 buffer contains data =2 2 buffers contain data =3 3 buffers contain data
D[25]	NODATA flag (response to event request) If NODATA = 0 there is no data corresponding to the requested event.
D[24]	=EOB (end of buffer) EOB = 0, the word currently read is the last belonging to the current event.
D[23]	Event number flag. If D[23]=1 the word currently read is an event number.

This flags are read addressing the board with a read cycle (ANY TDC internal address).

4.2 Initialization Procedure

At power up, the TDC chip registers are set as follows:

DLL mode enabled;

TDC chip internal clock disabled;

On board CKOUT connector disabled;

Common Stop mode selected;

Falling edge recording;

TDC chip testmode flag enabled;

Sequential read selected.

The user has now to initialize the TDC chip internal clock, in order to enable the Gray counter and the data acquisition, and to select the appropriate board working mode, by writing into the TDC chip internal registers.

The following list contains the commands used to initialize the TDC board:

1. Disable clock
2. Set TDC main control register (enable test mode);
3. Set RANG register (time window);
4. Set CREG register (DLL mode);
5. Begin clock (BECK) command;
6. Clear (CLAL) command, or a pulse on the CLEAR input (board front panel);
7. Set channel masks.

8. Enable TDC chip internal clock, disable test mode and select working mode.

During the initialization procedure, the TDC chip should not receive hit or stop signals.

Channels not used should be masked, in order not to have noise problems during the hit time recording.

Example 1: procedure to initialize the TDC board in common start, DLL mode, sequential reading, 16 hits max recorded per event, 16 bits event counter range, rising edge enabled, all channels enabled:

1. MAIN[15:0] = 0xF53F (testmode flag enabled, internal clock disabled);
2. RANG[15:0] = 0x3B (example: start gate shorter than 4 μs);
3. CREG[15:0] = 0x0 (clock setting);
4. BECK command;
5. CLAL command;
6. MASKL[15:0] = 0xFFFF; (enable channels 15-0)
7. MASKH[15:0] = 0xFFFF;(enable channels 31-16)
8. MAIN[15:0] = 0xF11F (testmode flag disabled, internal clock enabled).

Example 2: procedure to initialize the TDC board in common stop, DLL mode, sequential reading, 10 hits max recorded per event, time window of 2 μs, both edges enabled, 16 low channels enabled:

1. MAIN[15:0] = 0xF779 (testmode flag enabled, internal clock disabled);
2. RANG[15:0] = 0x3A (time window);
3. CREG[15:0] = 0x0 (clock setting);
4. BECK command;
5. CLAL command;
6. MASKL[15:0] = 0xFFFF; (enable channels 15-0)
7. MASKH[15:0] = 0x0000;(disable channels 31-16)
8. MAIN[15:0] = 0xF359 (testmode flag disabled, internal clock enabled).

4.3 Readout routine

When the board registers are set, the user can begin the data acquisition. The HIT and STOP signals can be sent to the front panel inputs, and the data readout can begin via VME read commands. These are the steps the readout routine should execute for each event:

```
Wait for EMBUF to go high;
LOOP
    VME read cycle at address (VMEBASE + 0x00)
    check if event number /*D[23]*/
    decode data
UNTIL EOB = 0 /*D[24] - end of event*/
```

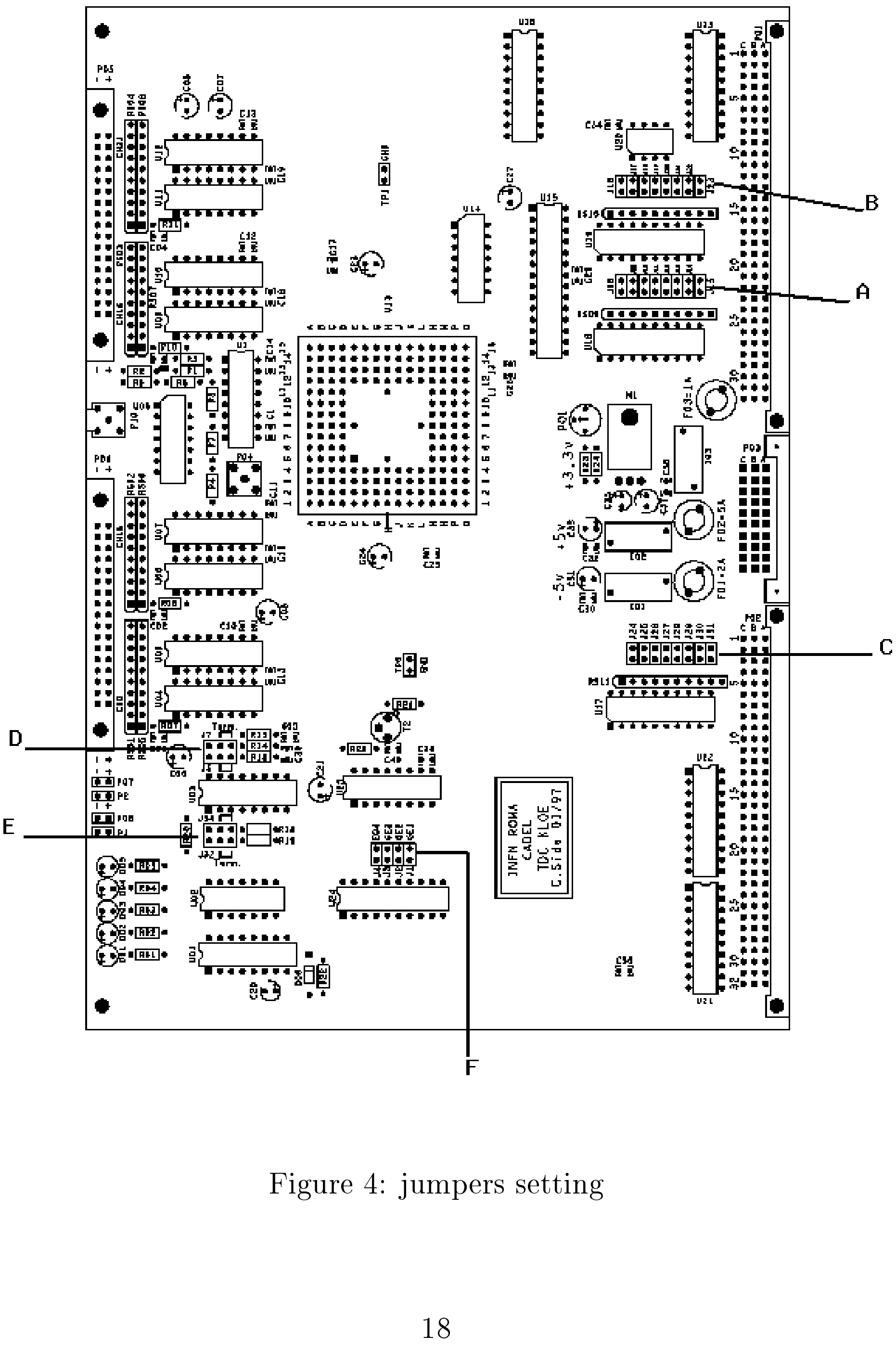

Figure 4: jumpers setting