High Pt Pretrigger Electronics

Pretrigger Board

H. Riege, J. Schütt, R. van Staa
Universität Hamburg

II. Institut für Experimentalphysik
May 2001

Content

Introduction

The High-Pt Pretrigger Electronics is built of three different boards:

- The Link Board **LB**, which is located near the detector and provides via optical fibres the fast data transfer to the main trigger logic.
- The Pretrigger Board **PB**, which searches for coincidence pattern as trigger candidates and combines the involved pad information of three detector layers to data sets, which are transmitted to the third board,
- The Message Generator **MG**, which transforms the received data to messages, which are accepted by the Track Finding Unit **TFU** of the HERA-B First Level Trigger System.

This Manual describes the Pretrigger Board of the High-Pt Pretrigger System. In the first chapter the logic of the board is shortly presented on the block diagram level. The next chapters report the control, programming and test facilities provided via VME. Finally an overview of the VME instructions, implemented on the board, is given.

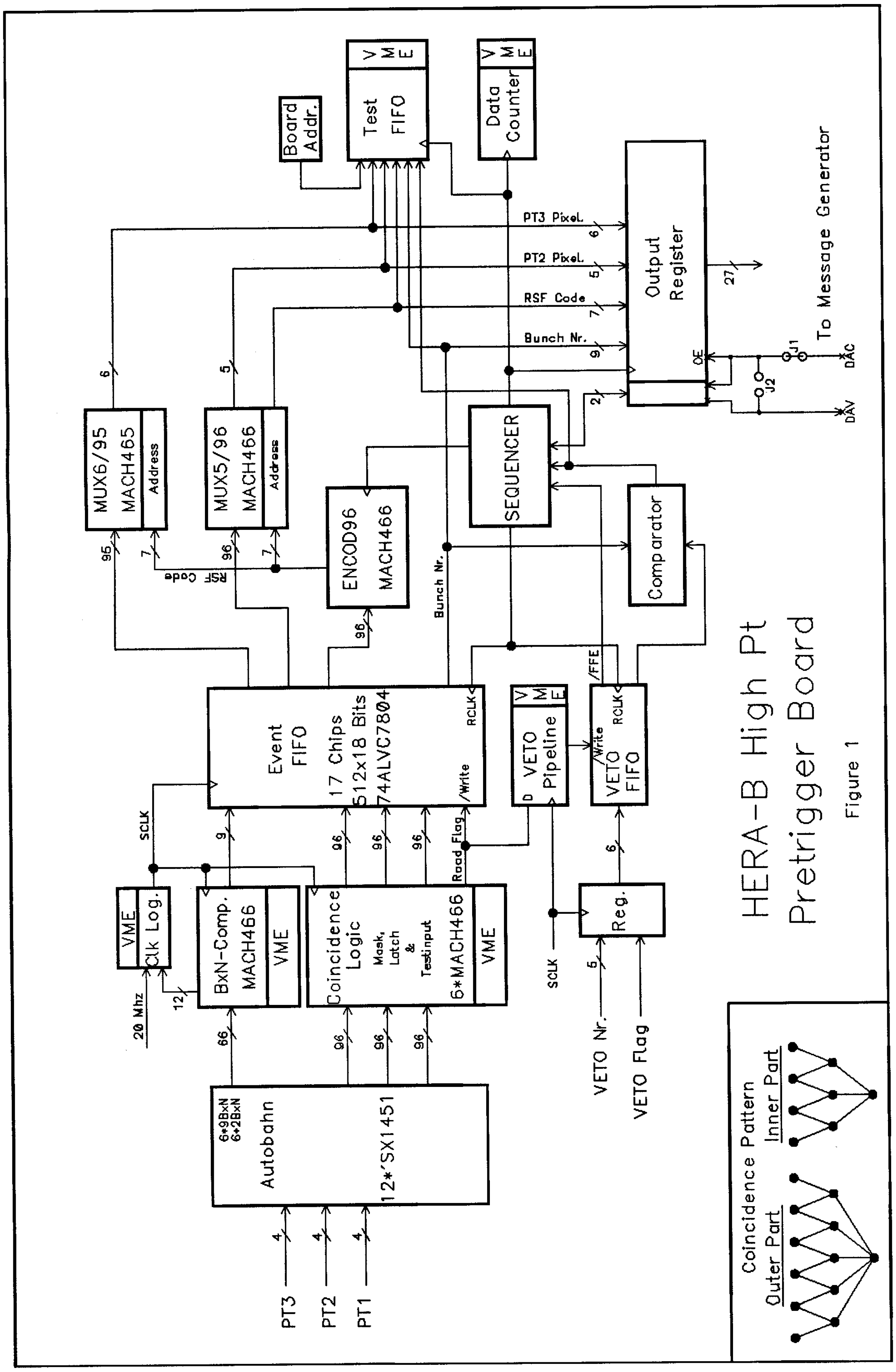
Logic Description

The High-Pt Pretrigger Board **PTB** is able to process during half a bunch crossing interval (48 nsec) the pixel data of a complete row of all three detector layers. Since the maximum number of pixels per row is 96, one has to transfer 288 detector bits together with redundant bunch number information to the **PTB** every 48 nsec. Table 1 shows, how these bits are allocated to the 12 optical channels, which are used for data transfer between the Link Boards and Pretrigger Boards.

Link Channel	Detector Pixel Bits	Bunch Number Bits	Detector Part
LA1	DLA0..DLA29	BXN0..BXN1	Layer1, left Half
LA2	DLA30..DLA47	BXN0..BXN8	Layer1, left Half
LB1	DLB0..DLB29	BXN0..BXN1	Layer2, left Half
LB2	DLB30..DLB47	BXN0..BXN8	Layer2, left Half
LC1	DLC0..DLC29	BXN0..BXN1	Layer3, left Half
LC2	DLC30..DLC47	BXN0..BXN8	Layer3, left Half
RA1	DRA0..DRA29	BXN0..BXN1	Layer1, right Half
RA2	DRA30..DRA47	BXN0..BXN8	Layer1, right Half
RB1	DRB0..DRB29	BXN0..BXN1	Layer2, right Half
RB2	DRB30..DRB47	BXN0..BXN8	Layer2, right Half
RC1	DRC0..DRC29	BXN0..BXN1	Layer3, right Half
RC2	DRC30..DRC47	BXN0..BXN8	Layer3, right Half

Table1: Allocation of Data Bits to Optical Link Channels

Here **BXN0** is the Cycle Bit, which distinguishes between two data sets, transmitted during one bunch crossing cycle, **BXN1..BXN8** are the Bunch Number Bits, and **DXYn** (X=L,R; Y=A,B,C; n=0..47) are the pixel bits of the detector. It can be seen, that there are 6 Front End Clusters (**LA, LB, LC, RA, RB, RC**) with two link channels each.
It should be mentioned, that the Detector Pixel Numbers **DPN** are increasing from the centre of the detector to the left and right border, while the Logical Pixel Number **LPN** are counted from left to right, as can be seen in the Allocation Table in the appendix. That table provides the allocation between the **LPN**, which is used to encode the **PTB** output data sets, and the **DPN**.
The stream of data bits coming in on 12 optical fibres (see left side of fig. 1) is serial-to-parallel converted by 12 Autobahn chips (Motorola). The 6*48 pixel bits are stored in 6 large CPLD's of type MACH466 (Lattice), while the 6*11 Bunch Number bits are transmitted to the Bunch Number Comparator, which checks the data transfer integrity, flags transfer errors and suppresses events with comparison failures. The Coincidence Logic and the Bunch Number Comparator are described in more detail in the following chapters.
The Coincidence Logic looks after predefined pattern of coincidences between pixels at corresponding rows of the three detector layers. Each pixel of the first layer, which is found to be the origin of a road pattern, is flagged by a Road Starting Flag **RSF**. If at least one **RSF** has been found, all **RSF**'s together with all pixel bits of the other layer rows and the Bunch Number are stored in the Event FIFO of 297 bits width and 512 events depth.
At the other side of the Event FIFO there is a Sequencer, which sequentially encodes all **RSF**'s of one event and forms data sets, which contain the **RSF** code, the corresponding pixel bits of the second and third detector layer and the 9 bit Bunch Number information. So it reads one event from the FIFO and transmits the **RSF**'s to an priority encoder ENCOD96, which at first encodes the most significant Road Starting Flag. That code is used to address two multiplexers MUX5/96 and MUX6/95, which select the associated pixel out of 96 bits of the second layer and 95 bits of the third layer respectively. Finally the complete data set is



HERA-B High Pt Pretrigger Board

Figure 1

stored in an Output Register and written to a Test FIFO. Encoder, Multiplexer and Output Register are forming three stages of a pipeline, so that it is possible to provide a new data set every 96 nsec.

For synchronisation purposes the system clock with a period of 48 nsec is derived from the Cycle Bits of all 12 links. The clock generation and control is covered in greater detail by a later chapter.
The VETO information, consisting of the five least significant Bunch Number bits and a VETO flag for those events, which have to be suppressed, is distributed by the Message Generator to all connected Pretrigger Boards. It is assumed, that the VETO information arrives at the Pretrigger Board later than the corresponding detector data. That can be obtained by selecting an appropriate delay in VETO Distribution Box (see Manual of that Box). The difference in arrival time then can be compensated in steps of 48 nsec by programming the length of the VETO Pipeline via VME.
The Road Flag, which writes the event data into the Event FIFO, has to be delayed by the VETO Pipeline so, that it writes the correct VETO Bunch Number into the VETO FIFO. Since the Sequencer observes the 'Not Empty' flag of the VETO FIFO, the event data are read only after writing the VETO Bunch Number to the VETO FIFO. The Event Bunch Number and the VETO Bunch Number then can be compared. If they are equal and if the VETO Flag is set, the event is suppressed. The procedure of finding the correct VETO Pipeline length is described in a later chapter.
In order to be able to test the logic of the board, there are Test Registers implemented, which cover the complete input range of 288 bits. The Test Register bits are numbered in terms of logical pixel bits. After writing a certain pattern to the Test Register, an evaluation cycle can be started by means of a dedicated VME command. The resulting output data sets then cab read back from the Test FIFO for comparison with the expected values.
Finally there is a Data Counter, which counts the number of data sets, written to the Output Register and to the Test FIFO. It can be read by VME for monitoring purposes.
The data transfer between Pretrigger Board and Message Generator is organised as a two wire Handshake. After writing a data set into the Output Register the **PTB** sets a **DAV** flag indicating, that new data are available. As a response, the Message Generator activates a **DAC** flag, which opens the **PTB** output port to the data bus and resets **DAV**. Finally the Message Generator writes the incoming data into it's Input Register and releases **DAC**.
Additionally an automatic Handshake has been implemented in order to be able to operate the Pretrigger Board without Message Generator. There are two jumpers **J1** and **J2**, which select between these two operation modes (see appendix).

Coincidence Logic

The upper part of Fig. 2 shows the input circuit of the Coincidence Logic CPLD's. At first each pixel bit (**DETECTOR_DATA**) is registered by a clock signal, which is the inverted **Full** flag of the corresponding Autobahn chip.
After passing a Multiplexer, which selects between the Input Register (**TESTMODE**=0) and the Test Register (**TESTMODE**=1), the pixel bit is masked by the content of the Mask Register **Mask0** (**Cycle**=0) or **Mask1** (**Cycle**=1), before it is stored in the Buffer Register. That happens every 48 nsec for all 288 pixel bits.
Register **Mask1** is used as Test Register, if **TESTMODE**=1. By means of a VME command, **START_TEST** can be set high for a short time. Then the content of the Test Register is written to the Buffer Register by the next **SCLK** pulse. So a data processing cycle is started, but only one, because the Test Register is cleared immediately by the $Q^{\sim}$-Output of the Buffer FlipFlop's.
The Buffer outputs are connected to the Coincidence Logic, as indicated in the lower part of Fig. 2. If a pixel of layer 1 is found to be the origin of a predefined road pattern, a Road Starting Flag is set for that bit, while the pixel information of the other layers are remaining

unchanged. Further 48 nsec later the 288 output bits of the CPLD’s are written to the event FIFO, if at least one **RSF** has been found.

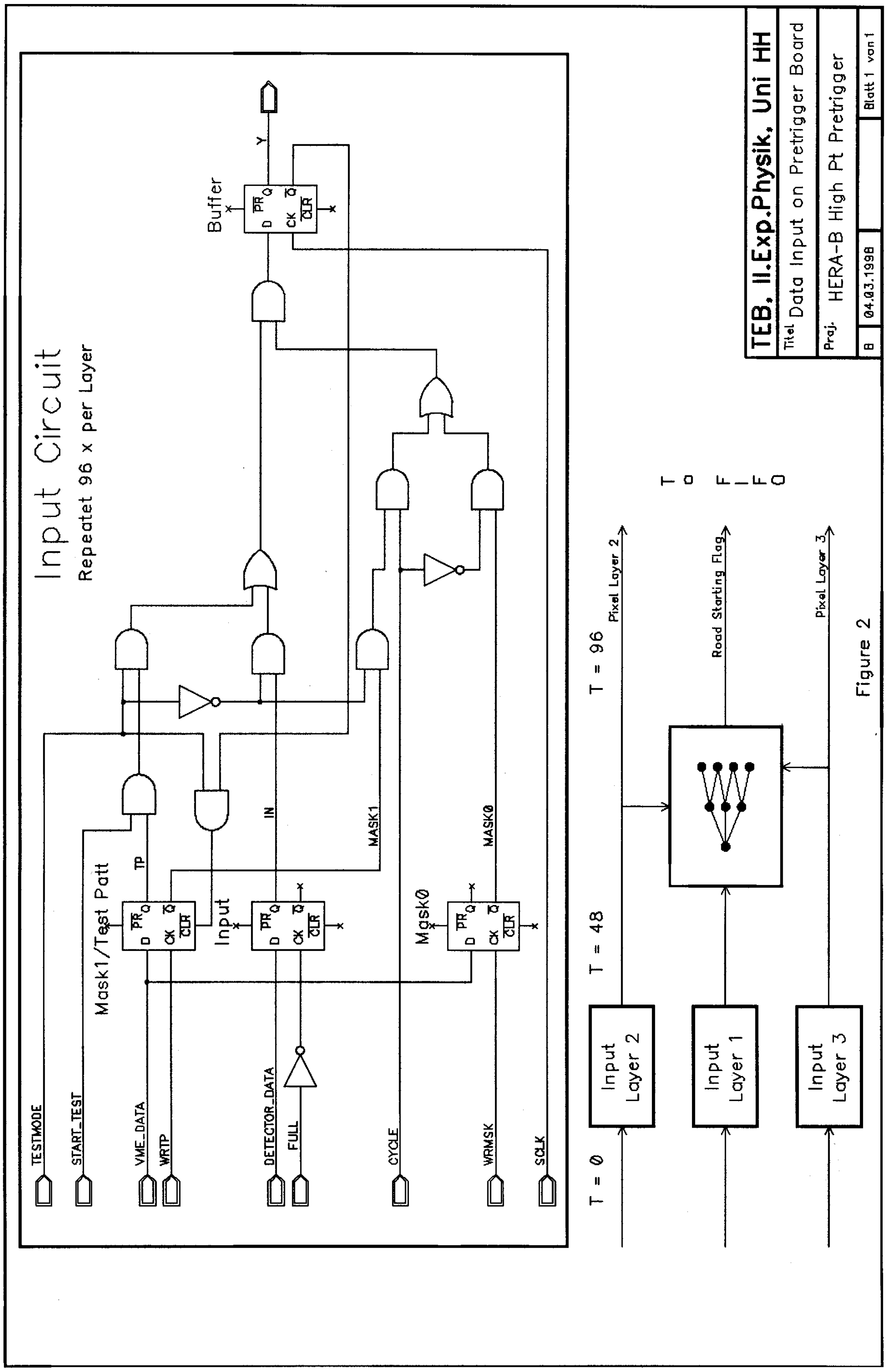


Figure 2

Bunch Number Comparison

Fig. 3 shows a block diagram of the Bunch Number comparison. For each Front End Cluster the Cycle Bit and the least significant Bunch Number bit, transmitted by the first link channel, and the Cycle Bit and the complete 8 bit Bunch Number, arriving at the second link channel (see Table 1), are clocked into an Input Register by the inverted **Full** flag of the corresponding Autobahn chip.

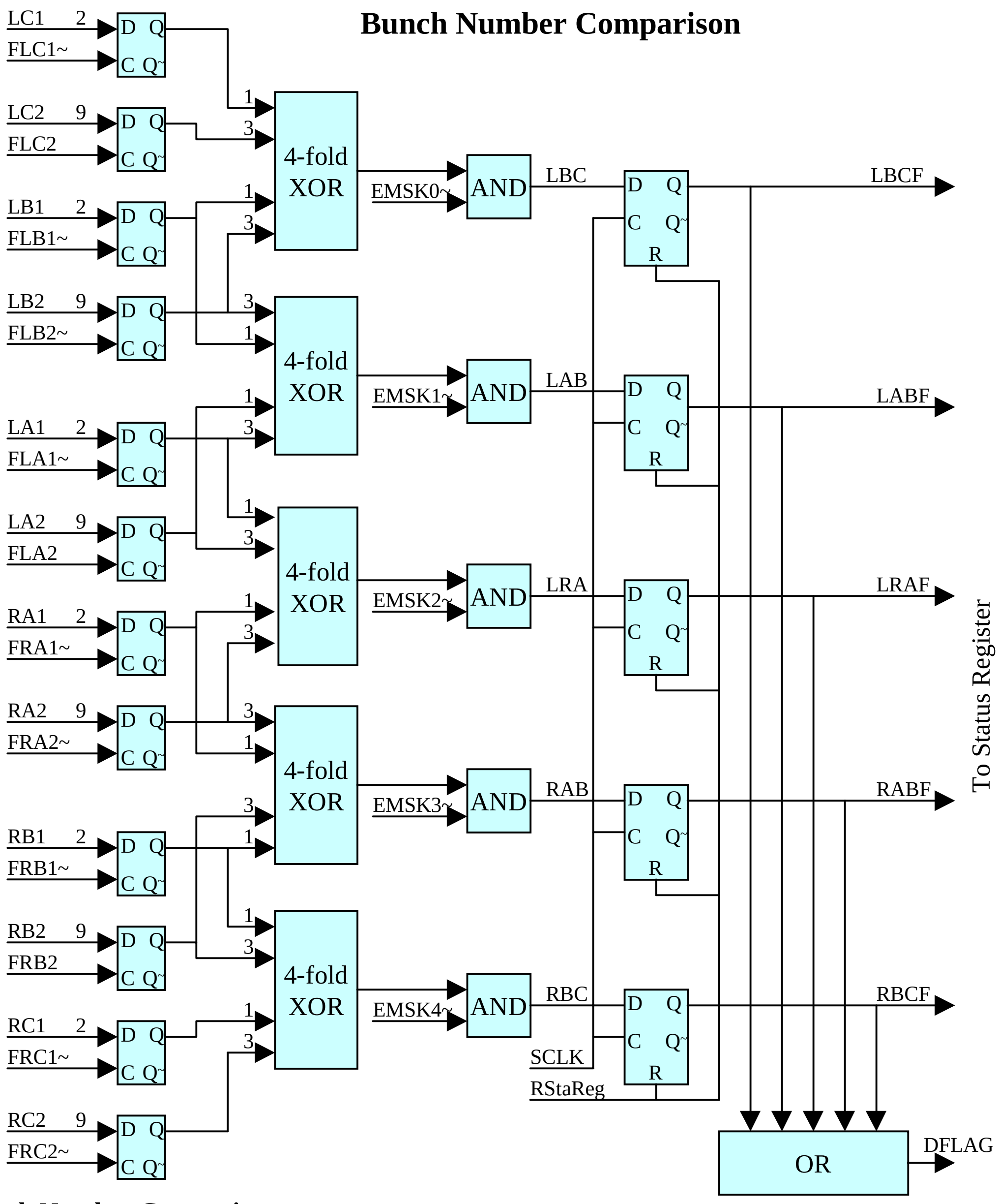


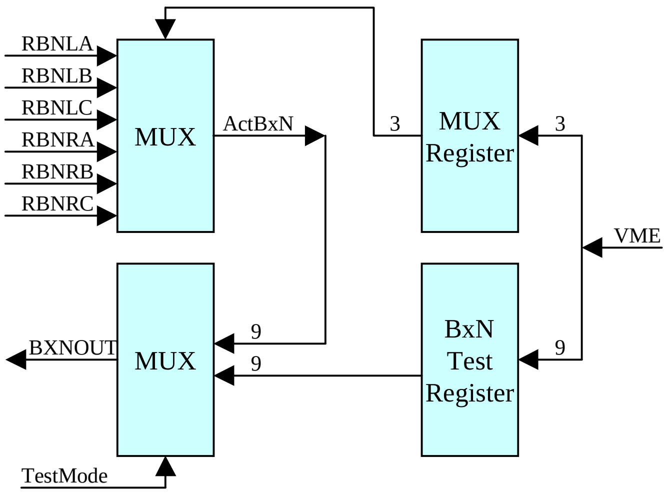
Fig.3: Bunch Number Comparison

One Bunch Number bit of the first link and three Bunch Number of the second link are used for comparison, which is done by XOR gates. The following five comparisons are implemented:

Layer 3, Left Half	-	Layer 2, Left Half
Layer 2, Left Half	-	Layer 1, Left Half
Layer 1, Left Half	-	Layer 1, Right Half
Layer 1, Right Half	-	Layer 2, Right Half
Layer 2, Right Half	-	Layer 3, Right Half

The resulting error flags, which can be masked by setting **EMSKN**$^{\sim}$=0, are registered by the system clock **SCLK** and then connected to the Status Register, where they are stored for monitoring and error diagnosis. The logical OR of all unmasked error flags is the Difference Flag **DFLAG**, which is used for event suppression.

For Bunch Number selection there is a Multiplexer MUX, which selects one of the six Input Registers as data source for the actual Bunch Number, which is transmitted to the Event FIFO.



In Test Mode (**TESTMODE**=1) the actual Bunch Number is given by the content of the BXN Test Register, which can be accessed via VME as well as the MUX Register and the BXN Error Mask Register.

Fig.4: Bunch Number Selection

Event Suppression by VETO

A certain event will only be suppressed by the VETO flag, if the Sequencer detects, that Event Bunch Number and VETO Bunch Number are equal (see Fig. 1). Therefore the event data have to be delayed such, that both numbers are arriving at the same time at the Sequencer comparator.

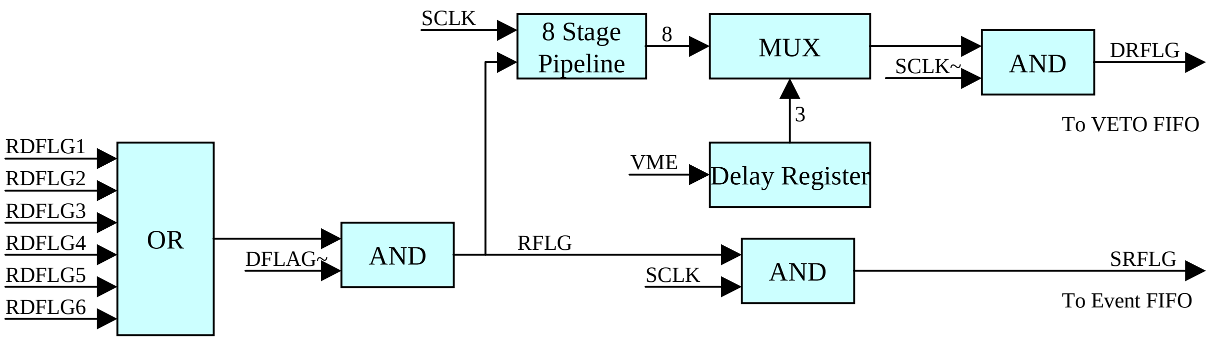


Fig.5: VETO Pipeline

For that purpose an extra VETO FIFO for the VETO Bunch Number has been implemented. The Road Flag **RFLG**, which writes the event data of a certain Bunch Number into the Event FIFO, is delayed so, that it can write into the VETO FIFO the later arriving VETO Bunch Number of the same value. Then the same value of Event Bunch Number and VETO Bunch Number is stored at the same stage of both FIFO's.

Fig.5 shows, how this programmable delay of the Road Flag has been implemented. **RFLG** is the logical OR of all Road Flags, coming from the different Coincidence Logic CPLD's. It is masked by the Difference Flag **DFLAG** of the Bunch Number Comparison and writes the event data into the Event FIFO. Furthermore it is shifted through a pipeline by the System Clock. By means of the Delay Register, one of the eight pipeline outputs is selected, providing a delay in steps of 48 nsec. The Delayed Road Flag **DRFLG** then is used as input clock for the VETO FIFO.

Before using the VETO facility one has to set up the correct delay time. For that purpose one at first has to disable the VETO function by setting the third bit of the Command Register to **VTEN**=0. Then for different delay values one has to take some data and to monitor the **VETO**˜ flag, which together with the data sets is stored in the Test FIFO. If the correct delay has been selected, one should observe some events with **VETO**$^{\sim}$=0, while for incorrect delay values always **VETO**$^{\sim}$=1 is detected.

System Clock Generation

In order to be able to synchronise the incoming data on the Pretrigger Board, the boards system clock **SCLK** is derived from the Cycle Bits, arriving at each Autobahn chip. Fig.6 shows a block diagram.

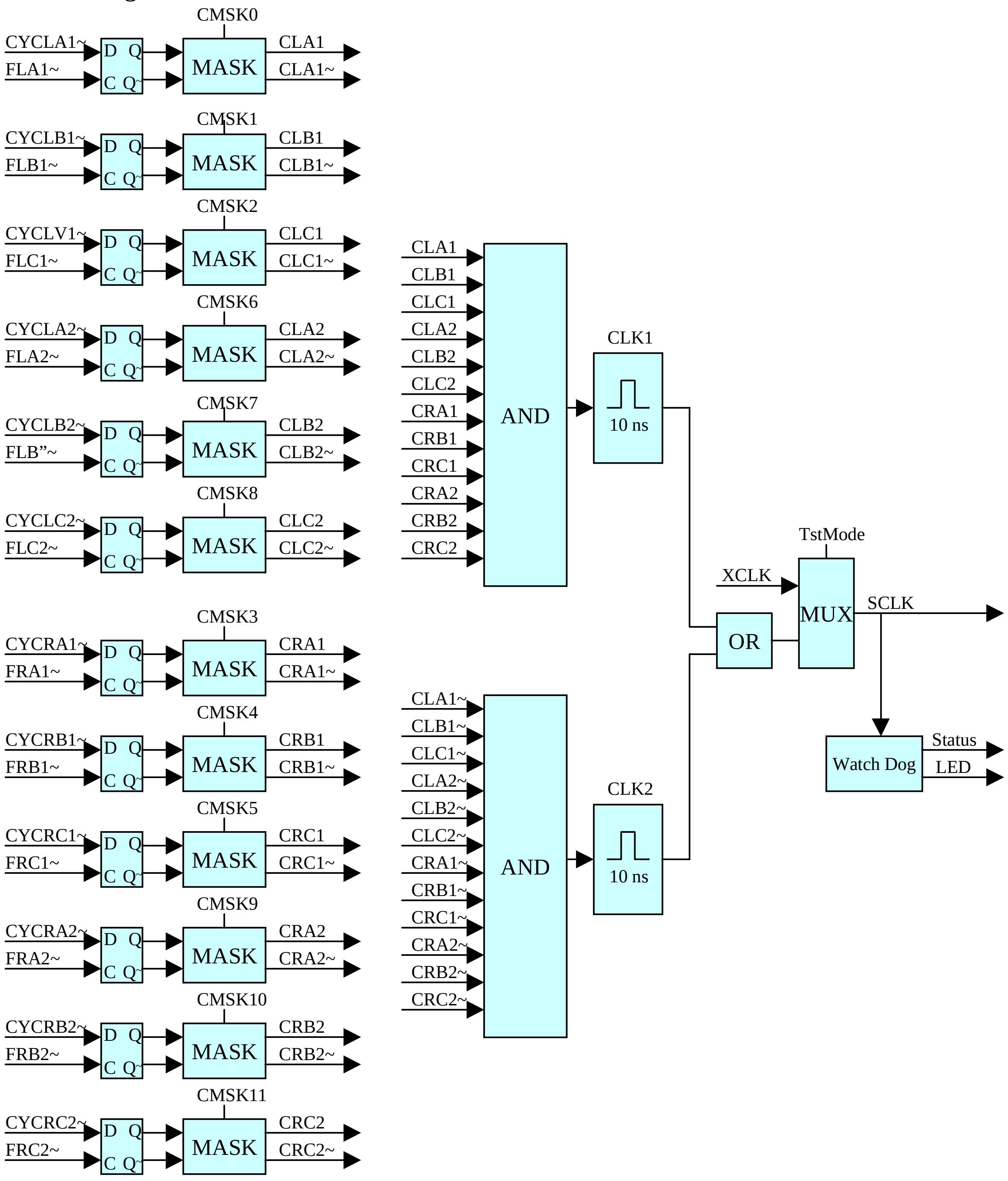


Fig.6: System Clock Generation

One pulse **CLK1** of the system clock is formed by the logical AND of the non inverted register outputs, while 48 nsec later a second pulse **CLK2** is given by the logical AND of the inverted register outputs. **CLK1** and **CLK2** are clipped to a length of 10 nsec each and then combined by a logical OR. Finally the resulting System Clock with a period of 48 nsec passes a Multiplexer, which in Test Mode selects a 20 MHz oscillator clock **XCLK**.

The propagation delay between the Input Register and the **SCLK** output is about 18 nsec. Therefore a spread in arrival time of the order of 30 nsec is tolerable.
If one **Full** flag or one Cycle Bit is missing, the System Clock fails. Therefore a Watch Dog circuit has been provided, which monitors the System Clock and, in case of a failure, sets an error flag in the Status Register and switches on a LED on the front panel of the board. By means of the Cluster Mask Register bits **CMASKn**, one can mask the defective link and return the System Clock to operation.

Digital to Analogue Converters
Each optical receiver on the board has to be adjusted individually by setting an offset voltage in the range between 0V and 2.5V. For remote control of that voltage 12 Digital to Analogue Converters (DAC) of 8 bit resolution have been implemented. They can be programmed by writing the DAC number and the DAC data byte **DB** to the DAC Register via VME.
The resulting output voltage **V** can be calculated by the formula

$$V = 2.5V * DB / 255.$$

The optimal offset voltage for a certain receiver can be found by the following procedure. At first one deselects all other links by setting the appropriate bits in the Clock Mask Register. Then by stepping through the DAC range and observing the Watch Dog response, one should find an offset voltage range, at which the link under test works stable. The centre of that range should be selected as working point.

VME Instructions

The VME Interface of the board supports **Short Supervisory Access** and **Short Non Privileged Access** (Address Modifier $29 and $2D). The address lines are completely decoded. The six most significant bits **A15..A10** are occupied by the Board Address BAD, which is determined by the slot location. The remaining 9 bits **A9..A1** are forming the address space of the board.

The following table provides a list of all instructions implemented. The second column gives the instruction address (hexadecimal notation), which has to be added to the Base Address of the board.

The Base Address can be calculated from the Board Address BAD by means of the following formula:

Base (Byte) Address = BAD * $400

The complete instruction (byte) address then is given by:

Instruct. Address = Base Address + Ad,

where **Ad** is given by the following table:

Instruction	Ad	A9..A7	A6	A5	A4	A3	A2	A1	Acc
General Clear	0	0	0	0	0	0	0	0	write
Read Status Register	0	0	0	0	0	0	0	0	read
Autobahn Clear	2	0	0	0	0	0	0	1	write
Write Command Register	4	0	0	0	0	0	1	0	write
Read Command Register	4	0	0	0	0	0	1	0	read
Clear Interrupt Flag	6	0	0	0	0	0	1	1	write
Clear Event Counter, Start Counting	8	0	0	0	0	1	0	0	write
Read Event Counter LSW, Stop Counting	8	0	0	0	0	1	0	0	read
Read Event Counter MSW	A	0	0	0	0	1	0	1	read
Clear Test FIFO	C	0	0	0	0	1	1	0	write
Read Test FIFO low	C	0	0	0	0	1	1	0	read
Read Test FIFO high	E	0	0	0	0	1	1	1	read
Start Test	E	0	0	0	0	1	1	1	write
Write PT1 1.Mask Register (B0..B15)	10	0	0	0	1	0	0	0	write
Read PT1 1.Mask Register (B0..B15)	10	0	0	0	1	0	0	0	read
Write PT1 2.Mask/Test Register (B0..B15)	12	0	0	0	1	0	0	1	write
Write PT2 1.Mask Register (B0..B15)	14	0	0	0	1	0	1	0	write
Read PT2 1.Mask Register (B0..B15)	14	0	0	0	1	0	1	0	read
Write PT2 2.Mask/Test Register (B0..B15)	16	0	0	0	1	0	1	1	write
Write PT3 1.Mask Register (B0..B15)	18	0	0	0	1	1	0	0	write
Read PT3 1.Mask Register (B0..B15)	18	0	0	0	1	1	0	0	read
Write PT3 2.Mask/Test Register (B0..B15)	1A	0	0	0	1	1	0	1	write
Write PT1 1.Mask Register (B16..B31)	20	0	0	1	0	0	0	0	write
Read PT1 1.Mask Register (B16..B31)	20	0	0	1	0	0	0	0	read
Write PT1 2.Mask/Test Register (B16..B31)	22	0	0	1	0	0	0	1	write
Write PT2 1.Mask Register (B16..B31)	24	0	0	1	0	0	1	0	write
Read PT2 1.Mask Register (B16..B31)	24	0	0	1	0	0	1	0	read
Write PT2 2.Mask/Test Register (B16..B31)	26	0	0	1	0	0	1	1	write
Write PT3 1.Mask Register (B16..B31)	28	0	0	1	0	1	0	0	write
Read PT3 1.Mask Register (B16..B31)	28	0	0	1	0	1	0	0	read
Write PT3 2.Mask/Test Register (B16..B31)	2A	0	0	1	0	1	0	1	write
Write PT1 1.Mask Register (B32..B47)	30	0	0	1	1	0	0	0	write
Read PT1 1.Mask Register (B32..B47)	30	0	0	1	1	0	0	0	read

Instruction	Ad	A9..A7	A6	A5	A4	A3	A2	A1	Acc
Write PT1 2.Mask/Test Register (B32..B47)	32	0	0	1	1	0	0	1	read
Write PT2 1.Mask Register (B32..B47)	34	0	0	1	1	0	1	0	write
Read PT2 1.Mask Register (B32..B47)	34	0	0	1	1	0	1	0	read
Write PT2 2.Mask/Test Register (B32..B47)	36	0	0	1	1	0	1	1	write
Write PT3 1.Mask Register (B32..B47)	38	0	0	1	1	1	0	0	write
Read PT3 1.Mask Register (B32..B47)	38	0	0	1	1	1	0	0	read
Write PT3 2.Mask/Test Register (B32..B47)	3A	0	0	1	1	1	0	1	write
Write PT1 1.Mask Register (B48..B63)	40	0	1	0	0	0	0	0	write
Read PT1 1.Mask Register (B48..B63)	40	0	1	0	0	0	0	0	read
Write PT1 2.Mask/Test Register (B48..B63)	42	0	1	0	0	0	0	1	write
Write PT2 1.Mask Register (B48..B63)	44	0	1	0	0	0	1	0	write
Read PT2 1.Mask Register (B48..B63)	44	0	1	0	0	0	1	0	read
Write PT2 2.Mask/Test Register (B48..B63)	46	0	1	0	0	0	1	1	write
Write PT3 1.Mask Register (B48..B63)	48	0	1	0	0	1	0	0	write
Read PT3 1.Mask Register (B48..B63)	48	0	1	0	0	1	0	0	read
Write PT3 2.Mask/Test Register (B48..B63)	4A	0	1	0	0	1	0	1	write
Write PT1 1.Mask Register (B64..B79)	50	0	1	0	1	0	0	0	write
Read PT1 1.Mask Register (B64..B79)	50	0	1	0	1	0	0	0	read
Write PT1 2.Mask/Test Register (B64..B79)	52	0	1	0	1	0	0	1	write
Write PT2 1.Mask Register (B64..B79)	54	0	1	0	1	0	1	0	write
Read PT2 1.Mask Register (B64..B79)	54	0	1	0	1	0	1	0	read
Write PT2 2.Mask/Test Register (B64..B79)	56	0	1	0	1	0	1	1	write
Write PT3 1.Mask Register (B64..B79)	58	0	1	0	1	1	0	0	write
Read PT3 1.Mask Register (B64..B79)	58	0	1	0	1	1	0	0	read
Write PT3 2.Mask/Test Register (B64..B79)	5A	0	1	0	1	1	0	1	write
Write PT1 1.Mask Register (B80..B95)	60	0	1	1	0	0	0	0	write
Read PT1 1.Mask Register (B80..B95)	60	0	1	1	0	0	0	0	read
Write PT1 2.Mask/Test Register (B80..B95)	62	0	1	1	0	0	0	1	write
Write PT2 1.Mask Register (B80..B95)	64	0	1	1	0	0	1	0	write
Read PT2 1.Mask Register (B80..B95)	64	0	1	1	0	0	1	0	read
Write PT2 2.Mask/Test Register (B80..B95)	66	0	1	1	0	0	1	1	write
Write PT3 1.Mask Register (B80..B95)	68	0	1	1	0	1	0	0	write
Read PT3 1.Mask Register (B80..B95)	68	0	1	1	0	1	0	0	read
Write PT3 2.Mask/Test Register (B80..B95)	6A	0	1	1	0	1	0	1	write
Write BXN MUX Register	70	0	1	1	1	0	0	0	write
Read BXN MUX Register	70	0	1	1	1	0	0	0	read
Write BXN Error Mask Register	72	0	1	1	1	0	0	1	write
Read BXN Error Mask Register	72	0	1	1	1	0	0	1	read
Write Delay Register	76	0	1	1	1	0	1	1	write
Read Delay Register	76	0	1	1	1	0	1	1	read
Write BXN Test Register	78	0	1	1	1	1	0	0	write
Read BXN Test Register	78	0	1	1	1	1	0	0	read
Write Clock Mask Register	7A	0	1	1	1	1	0	1	write
Read Clock Mask Register	7A	0	1	1	1	1	0	1	read
Write DAC Register	7C	0	1	1	1	1	1	0	write
Read DAC Register	7C	0	1	1	1	1	1	0	read

The instructions are shortly described in the following:

General Clear ($00 W)
This instruction resets the State Machines on the board to a defined Ground State and clears the BXN MUX Register, the BXN Error Mask Register, the Delay Register, the BXN Test Register and the Clock Mask Register.

Autobahn Clear ($02 W)
This instruction resets the Autobahn chips and configures them to 32 bit mode.

Read Status Register ($00 R)

0	0	0	0	NTFF	NTFE	NEFF	NEFE	0	WDOG	LR1D	R23D	R12D	L23D	L12D	INT
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

INT:	Interrupt Request FlipFlop
	It is set by the logical equation:
	(LR1D+R23D+R12D+L23D+L12D)&IEN1 + WDOG&IEN2 + NEFF˜&IEN3
	where IEN1, IEN2, IEN3 are the Interrupt Mask Bits, set in the Command Register
L12D:	0: No BXN Difference between left half of PT1 and PT2
	1: BXN Difference between left half of PT1 and PT2 detected
L23D:	0: No BXN Difference between left half of PT2 and PT3
	1: BXN Difference between left half of PT2 and PT3 detected
R12D:	0: No BXN Difference between right half of PT1 and PT2
	1: BXN Difference between right half of PT1 and PT2 detected
R23D:	0: No BXN Difference between right half of PT2 and PT3
	1: BXN Difference between right half of PT2 and PT3 detected
LR1D:	0: No BXN Difference between left half and right half of PT1
	1: BXN Difference between left half and right half of PT1detected
WDOG:	0: System Clock is working
	1: System Clock is not working
NEFE:	0: Event FIFO is empty
	1: Event FIFO is not empty
NEFF:	0: Event FIFO is full
	1: Event FIFO is not full
NTFE:	0: Test FIFO is empty
	1: Test FIFO is not empty
NTFF:	0: Test FIFO is full
	1: Test FIFO is not full

Bits 1-6 are cleared after reading the Status Register. The logical OR of bits 1-5 is indicated by a front panel LED as well as the states of **WDOG** and **INT**.

Write/Read Command Register ($04 W/R)

IL3	IL2	IL1	IEN3	IEN2	IEN1	0	0	0	MRD3	MRD2	MRD1	MRD0	VTEN	TSTM	RUN
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RUN:	0: Data Processing on the board is disabled.
	1: Data Processing on the board is enabled.
TSTM:	0: Normal Data Acquisition Mode.
	1: Test Mode.
VTEN:	0: Data suppression by VETO Number is not enabled.
	1: Data suppression by VETO Number is enabled.
MRD3..MRD0:	Maximum number of roads per event.
	MRD3..MRD0=0: Number of roads per event is not limited.
IEN1:	0: BXN Difference as interrupt source is disabled.
	1: BXN Difference as interrupt source is enabled.
IEN2:	0: System Clock failure as interrupt source is disabled.
	1: System Clock failure as interrupt source is enabled.
IEN3:	0: Event FIFO Full flag as interrupt source is disabled.
	1: Event FIFO Full flag as interrupt source is enabled.

IL1IL1..IL3: Interrupt Level.

Clear Interrupt Flag ($06 W)

The Interrupt Request FlipFlop is reset.

Clear Event Counter, Start Counting ($08 W)

That command resets the Event Counter and enables counting.

Stop Event Counter, Read LSW ($08 R)

EC15	EC14	EC13	EC12	EC11	EC10	EC9	EC8	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

EC15..EC0: 16 least significant bits of Event Counter.

This Command stops the Event Counter and reads the Least Significant Word (LSW). So reading the Event Counter has to be done by at first applying **Read Event Counter LSW** followed by **Read Event Counter MSW**.

Read Event Counter MSW ($0A R)

ECOF	x	x	x	EC27	EC26	EC25	EC24	EC23	EC22	EC21	EC20	EC19	EC18	EC17	EC16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

EC27..EC16:	12 most significant bits of Event Counter.
ECOF:	Event Counter Overflow

Clear Test FIFO ($0C W)

A dataless command to clear the Test FIFO

Read Test FIFO low ($0C R)

RSF6	RSF5	RSF4	RSF3	RSF2	RSF1	RSF0	BXN8	BXN7	BXN6	BXN5	BXN4	BXN3	BXN2	BXN1	CYC
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CYC:	Cycle Bit
BxN1..BxN8:	Bunch Number Bits
RSF0..RSF6:	Road Starting Flags.

That command reads the least significant part of the Test FIFO and is the first command in a complete read cycle.

Read Test FIFO high ($0E R)

/VETO	BDN3	BDN2	BDN1	BDN0	PIC5	PIC4	PIC3	PIC2	PIC1	PIC0	PIB4	PIB3	PIB2	PIB1	PIB0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PIB0..PIB4:	Pads of 2. Detector Layer.
PIC0..PIC5:	Pads of 3. Detector Layer.
BDN3..BDN0:	Board Number
/VETO:	0: The data set is flagged by a VETO 1: The data set is not flagged by a VETO This flag has been implemented in order to find the correct timing for the VETO Number Delay. If the timing is correct, then there should be some events found with /**VETO**=0, if **VTEN**=0 is set in the Command Register.

That command reads the most significant part of the Test FIFO and then shifts the next event data to the FIFO output port. So a Test FIFO read cycle has to be started with **Read Test FIFO low** and continued with **Read Test FIFO high**.

Start Test ($0E W)

That command starts a test cycle, if **RUN**=1 and **TSTM**=1 is set in the Command Register.

Write/Read 1. Mask Register

MA15	MA14	MA13	MA12	MA11	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

MAx:	0: The corresponding pixel input is active.
	1: The corresponding pixel input is forced to 0.

In Data Acquisition Mode (**TSTM**=0) this register is used as Mask Register for input data, received in the first data transmission cycle (Cycle Bit = 0). In Test Mode (**TSTM**=1) this register is used as Mask Register for the Test Pattern.

Write/Read 2. Mask Register / Test Pattern Register

TP15	TP14	TP13	TP12	TP11	TP10	TP9	TP8	TP7	TP6	TP5	TP4	TP3	TP2	TP1	TP0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

TPx:	0: The corresponding Test Bit is set to 0
	1: The corresponding Test Bit is set to 1

In Data Acquisition Mode (**TSTM**=0) this register is used as Mask Register for input data, received in the second data transmission cycle (Cycle Bit = 1). In Test Mode (**TSTM**=1) this register is used as Test Pattern Register, and the 1. Mask Register is used as Mask Register.

Write/Read BXN MUX Register ($70 W/R)

x	x	x	x	x	x	x	x	x	x	x	x	x	MU2	MU1	MU0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

MU2..MU0:	0: Bunch Number of PT1 left half is selected
	1: Bunch Number of PT2 left half is selected
	2: Bunch Number of PT3 left half is selected
	3: Bunch Number of PT1 right half is selected
	4: Bunch Number of PT2 right half is selected
	5: Bunch Number of PT3 right half is selected

The BXN MUX Register is only operational, if **TSTM**=0 (see Command Register). In Test Mode (**TSTM**=1) the Bunch Number is provided by the BXN Test Register. At power-on or after **General Clear** all bits are reset to 0.

Write/Read BXN Error Mask Register ($72 W/R)

x	x	x	x	x	x	x	x	x	x	x	EMA4	EMA3	EMA2	EMA1	EMA0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

EMA0:	0: The Error Bit **L23D** (see Status Register) is active.
	1: The Error Bit **L23D** (see Status Register) is masked.
EMA1:	0: The Error Bit **L12D** (see Status Register) is active.
	1: The Error Bit **L12D** (see Status Register) is masked.
EMA2:	0: The Error Bit **LR1D** (see Status Register) is active.
	1: The Error Bit **LR1D** (see Status Register) is masked.
EMA3:	0: The Error Bit **R12D** (see Status Register) is active.
	1: The Error Bit **R12D** (see Status Register) is masked.
EMA4:	0: The Error Bit **R23D** (see Status Register) is active.
	1: The Error Bit **R23D** (see Status Register) is masked.

At power-on or after **General Clear** all bits are reset to 0.
In Test Mode (**TSTM**=1) the register has to be set to $1F, because the input signals to the Bunch Number Comparison are not defined.

Write/Read Delay Register ($76 W/R)

x	x	x	x	x	x	x	x	x	x	x	x	DEL3	DEL2	DEL1	DEL0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DEL3..DEL0: 0: The Road Flag is delayed by 1 time step (48 nsec)
1: The Road Flag is delayed by 2 time steps
n: The Road Flag is delayed by (n+1) time steps ($0 \le n \le 15$)

At power-on or after **General Clear** all bits are reset to 0.

Write/Read BXN Test Register ($78 W/R)

x	x	x	x	x	x	x	x	BNT7	BNT6	BNT5	BNT4	BNT3	BNT2	BNT1	BNT0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

BNT7..BNT0: Bunch Number, which is used in Test Mode (**TSTM**=1, see Command Register)

At power-on or after **General Clear** all bits are reset to 0.

Write/Read Clock Mask Register ($7A W/R)

x	x	x	x	RC2	RB2	RA2	LC2	LB2	LA2	RC1	RB1	RA1	LC1	LB1	LA1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LA1:	0: 1.Link of left half of PT1 (LA1) is included in System Clock generation.
	1: 1.Link of left half of PT1 (LA1) is excluded from System Clock generation
LB1:	0: 1.Link of left half of PT2 (LB1) is included in System Clock generation.
	1: 1.Link of left half of PT2 (LB1) is excluded from System Clock generation
LC1:	0: 1.Link of left half of PT3 (LC1) is included in System Clock generation.
	1: 1.Link of left half of PT3 (LC1) is excluded from System Clock generation
RA1:	0: 1.Link of right half of PT1 (RA1) is included in System Clock generation.
	1: 1.Link of right half of PT1 (RA1) is excluded from System Clock generation
RB1:	0: 1.Link of right half of PT2 (RB1) is included in System Clock generation.
	1: 1.Link of right half of PT2 (RB1) is excluded from System Clock generation
RC1:	0: 1.Link of right half of PT3 (RC1) is included in System Clock generation.
	1: 1.Link of right half of PT3 (RC1) is excluded from System Clock generation
LA2:	0: 2.Link of left half of PT1 (LA2) is included in System Clock generation.
	1: 2.Link of left half of PT1 (LA2) is excluded from System Clock generation
LB2:	0: 2.Link of left half of PT2 (LB2) is included in System Clock generation.
	1: 2.Link of left half of PT2 (LB2) is excluded from System Clock generation
LC2:	0: 2.Link of left half of PT3 (LC2) is included in System Clock generation.
	1: 2.Link of left half of PT3 (LC2) is excluded from System Clock generation
RA2:	0: 2.Link of right half of PT1 (RA2) is included in System Clock generation.
	1: 2.Link of right half of PT1 (RA2) is excluded from System Clock generation
RB2:	0: 2.Link of right half of PT2 (RB2) is included in System Clock generation.
	1: 2.Link of right half of PT2 (RB2) is excluded from System Clock generation
RC2:	0: 2.Link of right half of PT3 (RC2) is included in System Clock generation.
	1: 2.Link of right half of PT3 (RC2) is excluded from System Clock generation

At power-on or after **General Clear** all bits are reset to 0.

Write DAC Register ($7C W/R)

x	x	x	x	DAC3	DAC2	DAC1	DAC0	DAT7	DAT6	DAT5	DAT4	DAT3	DAT2	DAT1	DAT0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DAT7..DAT0:	8 bit DAC Data
DAC3..DAC0:	DAC Number 0: DAC LA1 1: DAC LB1 2: DAC LC1 3: DAC LA2 4: DAC LB2 5: DAC LC2 6: DAC RA2 7: DAC RB2 8: DAC RC2 9: DAC RA1 10: DAC RB1 11: DAC RC1

After writing to that register, a state machine on the board serially shifts the 8 data bits into the selected DAC. The DAC Register cannot be read back.

Appendix

1) *Detector Pixel Numbers DPN and Logical Pixel Numbers LPN*

LPN	**DPN**	**Input Reg.**	**CPLD Pins**		
			PT1	**PT2**	**PT3**
0	L47	U90	3	54	163
1	L46		4	55	164
2	L45		5	56	165
3	L44		6	57	171
4	L43		13	58	172
5	L42		14	64	173
6	L41		15	65	174
7	L40		16	66	175
8	L39		17	67	191
9	L38		32	68	192
10	L37		33	86	193
11	L36		34	87	200
12	L35		35	88	201
13	L34		36	89	202
14	L33		42	96	203
15	L32		43	97	204
16	L31	U91	3	54	163
17	L30		4	55	164
18	L29		5	56	165
19	L28		6	57	171
20	L27		13	58	172
21	L26		14	64	173
22	L25		15	65	174
23	L24		16	66	175
24	L23		17	67	191
25	L22		32	68	192
26	L21		33	86	193
27	L20		34	87	200
28	L19		35	88	201
29	L18		36	89	202
30	L17		42	96	203
31	L16		43	97	204
32	L15	U100	3	54	163
33	L14		4	55	164
34	L13		5	56	165
35	L12		6	57	171
36	L11		13	58	172
37	L10		14	64	173
38	L9		15	65	174
39	L8		16	66	175
40	L7		17	67	191
41	L6		32	68	192
42	L5		33	86	193
43	L4		34	87	200
44	L3		35	88	201
45	L2		36	89	202
46	L1		42	96	203
47	L0		43	97	204

LPN	DPN	Input Reg.	CPLD Pins		
			PT1	PT2	PT3
48	R0	U101	3	54	163
49	R1		4	55	164
50	R2		5	56	165
51	R3		6	57	171
52	R4		13	58	172
53	R5		14	64	173
54	R6		15	65	174
55	R7		16	66	175
56	R8		17	67	191
57	R9		32	68	192
58	R10		33	86	193
59	R11		34	87	200
60	R12		35	88	201
61	R13		36	89	202
62	R14		42	96	203
63	R15		43	97	204
64	R16	U110	3	54	163
65	R17		4	55	164
66	R18		5	56	165
67	R19		6	57	171
68	R20		13	58	172
69	R21		14	64	173
70	R22		15	65	174
71	R23		16	66	175
72	R24		17	67	191
73	R25		32	68	192
74	R26		33	86	193
75	R27		34	87	200
76	R28		35	88	201
77	R29		36	89	202
78	R30		42	96	203
79	R31		43	97	204
80	R32	U111	3	54	163
81	R33		4	55	164
82	R34		5	56	165
83	R35		6	57	171
84	R36		13	58	172
85	R37		14	64	173
86	R38		15	65	174
87	R39		16	66	175
88	R40		17	67	191
89	R41		32	68	192
90	R42		33	86	193
91	R43		34	87	200
92	R44		35	88	201
93	R45		36	89	202
94	R46		42	96	203
95	R47		43	97	204

2) Backplane Description

a) Backplane Connector P1
Standard P1/J1 VME Connector

b) Backplane Connector P2
Standard P2/J2 VME Connector, Column b not used

Pin	a	b	b
1	BAD0		BAD1
2	BAD2		BAD3
3	BAD4		BAD5
4	GND		GND
5			
6	+3.3V		+3.3V
7	+3.3V		+3.3V
8	+3.3V		+3.3V
9			
10			
11	GND		GND
12	GND		GND
13	GND		GND
14	GND		GND
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26	GND		GND
27	GND		GND
28			
29			
30	+4V		+4V
31			
32			

BAD5..BAD0: Board Address Bits, corresponding to **A15..A10**

c) Backplane Connector P3
96 Pin Connector, connected to Backplane of type VME-J1

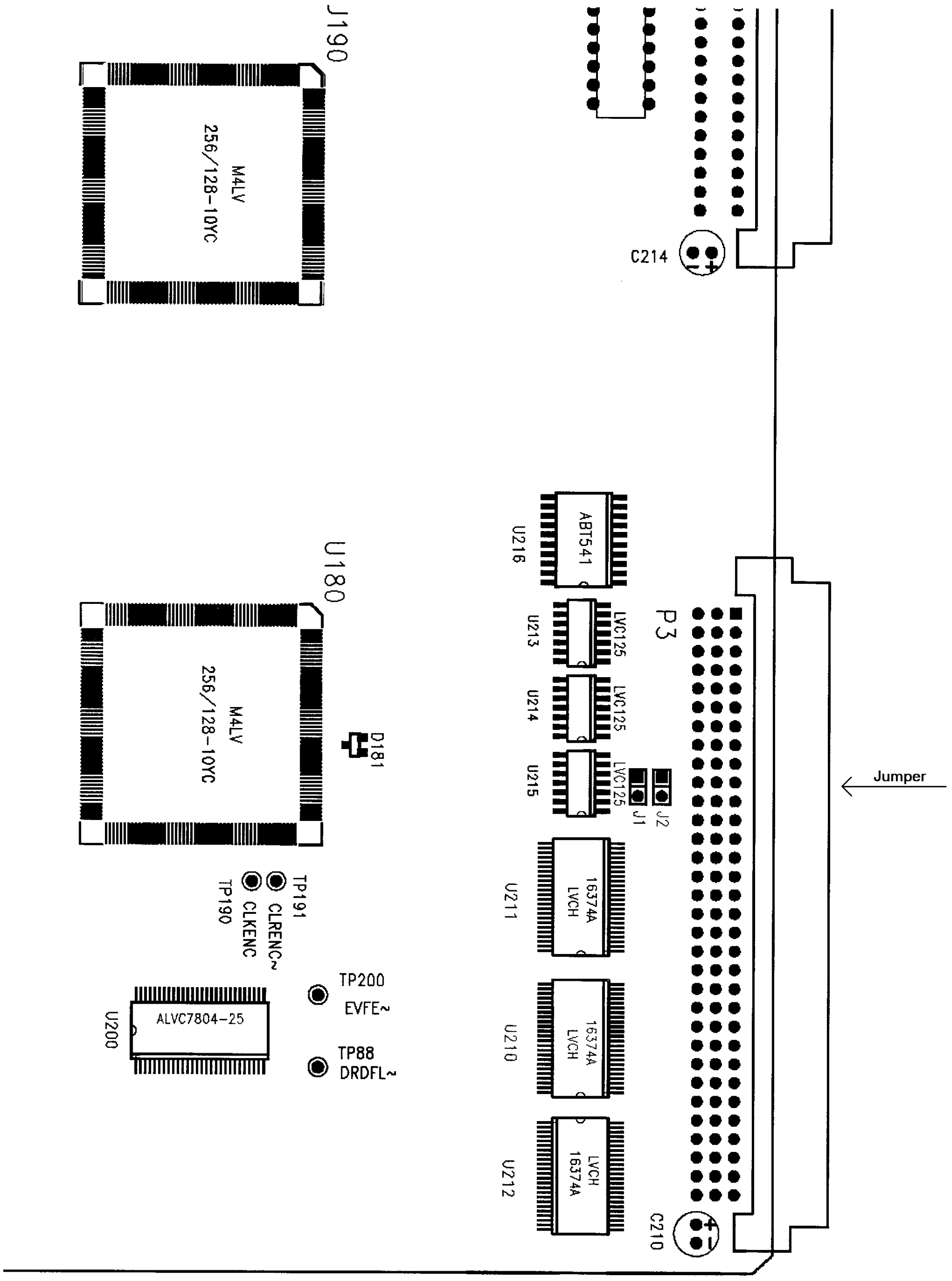
Pin	a	b	b
1	/DAV1		/DAC1
2	/DAV2		/DAC2
3	/DAV3		/DAC3
4	/DAV4		/DAC4
5	/DAV5		/DAC5
6	/DAV6		/DAC6
7	/DAV7		/DAC7
8	/DAV8		/DAC8
9	GND		GND
10	BCLK		
11	GND		
12	/DAV9		
13	/DAC9		BXN0
14	BXN1		BXN2
15	GND		BXN3
16		BXN4	BXN5
17	GND	BXN6	BXN7
18	BXN8	RSF0	RSF1
19	GND	RSF2	RSF3
20	RSF4	GND	RSF5
21		spare	RSF6
22		spare	PB0
23	PB1	GND	PB2
24	PB3		PB4
25	PC0		PC1
26	PC2		PC3
27	PC4		PC5
28	V0		V1
29	V2		V3
30	V4		VVAL
31			
32	+5V	+5V	+5V

/DAV9../DAV1:	Data Available Flags
/DAC9../DAC1:	Data Accepted Flags
BXN8..BXN1,BXN0:	Bunch Number, Cycle Bit
RSF6..RSF0:	Road Starting Flags
PIB4..PIB0:	Pixel of Layer 2
PIC5..PIC0:	Pixel of Layer 3
V4..V0,VVAL:	Veto Number Bits, Veto Number Valid Flag
BCLK:	Bunch Clock

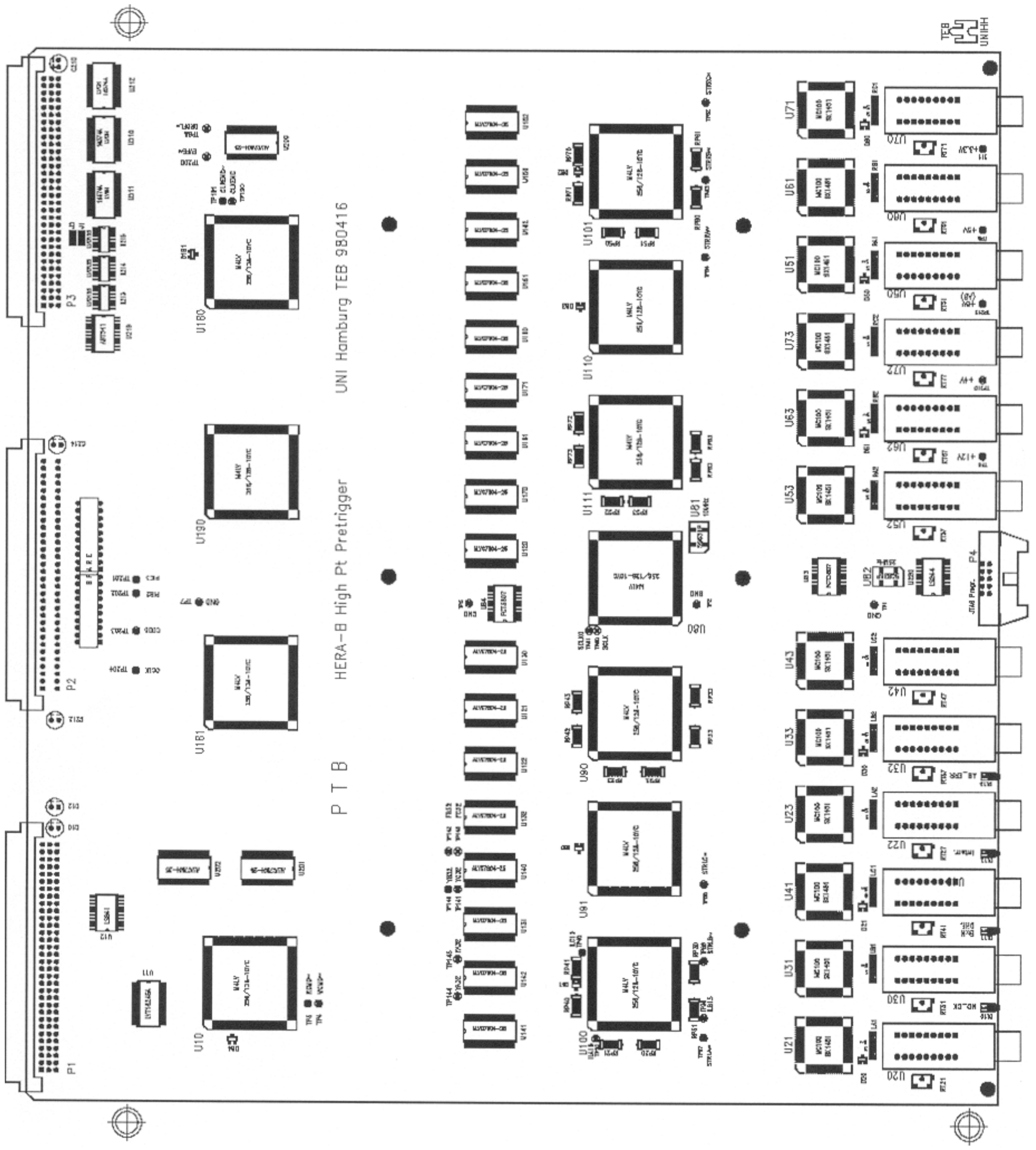
3) Jumpers

There are two jumpers **J1** and **J2**, which select between Normal Handshake (Operation with Message Generator) or Automatic Handshake (Operation without Message Generator). One and only one of the jumpers has to be closed:

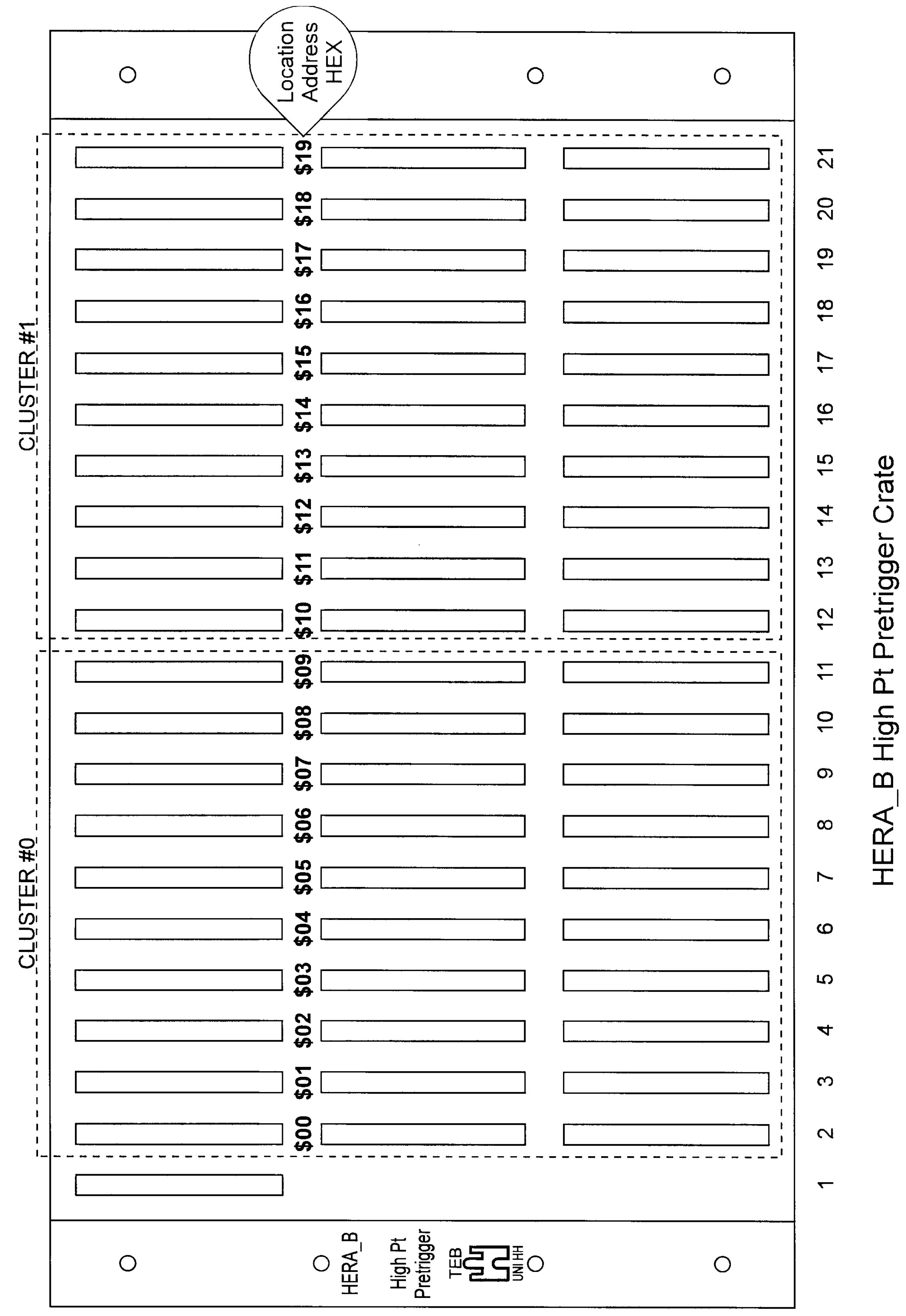
J1 closed:	Normal Handshake
J2 closed:	Automatic Handshake



4) *Board Layout*



6) *High-Pt Pretrigger Crate Backplane*



Location Address HEX
$19
$18
$17
$16
$15
$14
$13
$12
$11
$10
$09
$08
$07
$06
$05
$04
$03
$02
$01
$00
CLUSTER #1
CLUSTER #0
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
HERA_B High Pt Pretrigger Crate
HERA_B
High Pt Pretrigger
TEB
UNI HH