PSB
Pipelined Synchronising Buffer Module

9U-Version

H. Bergauer, K. Kastner, M. Padrta, A. Taurok

Oct-05

Version 1.2
with
PSB chip V0012
VME chip V1005
VME64x chip V1000B

1 PSB Module 9U description

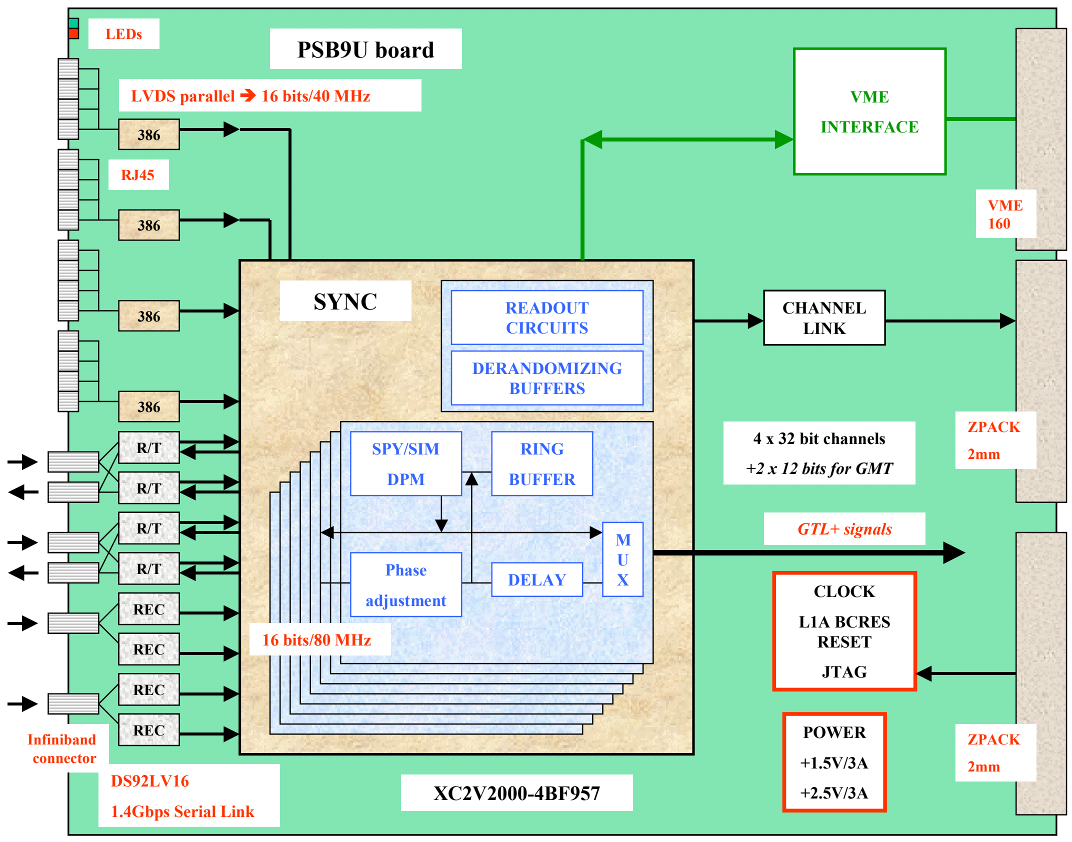


Figure 1 : PSB Overview

The PSB9U board receives 1.4 Gbps serial data from the Calorimeter trigger system, converts the serial bits into 16 bit words, synchronizes the words to the local clock, applies a programmable delay and sends the trigger objects as 80 MHz GTLp signals via the back-plane to the destination boards. Therefore each data channel contains the data bits of two time-multiplexed trigger objects.
The PSB9U boards are used by the Global Trigger as well as by the Global Muon Trigger.
Four Infiniband connectors forward the serial data to 8 DS92LV16 interface chips. After the serial to parallel conversion the 80 MHz data streams enter two Synchronization chips (=SYNC) where all the synchronization and monitoring logic is implemented.
The Synchronization Chip contains the over-sampling circuit that samples each word 4 times that means every 12.5/4 = 3.01 ns. The sample furthest from the data switching time is connected to the following programmable delay. After the delay the trigger words go through

a multiplexer and then as terminated GTLp signals to the back-plane. The multiplexer circuit allows sending test data instead of trigger data to the back-plane.

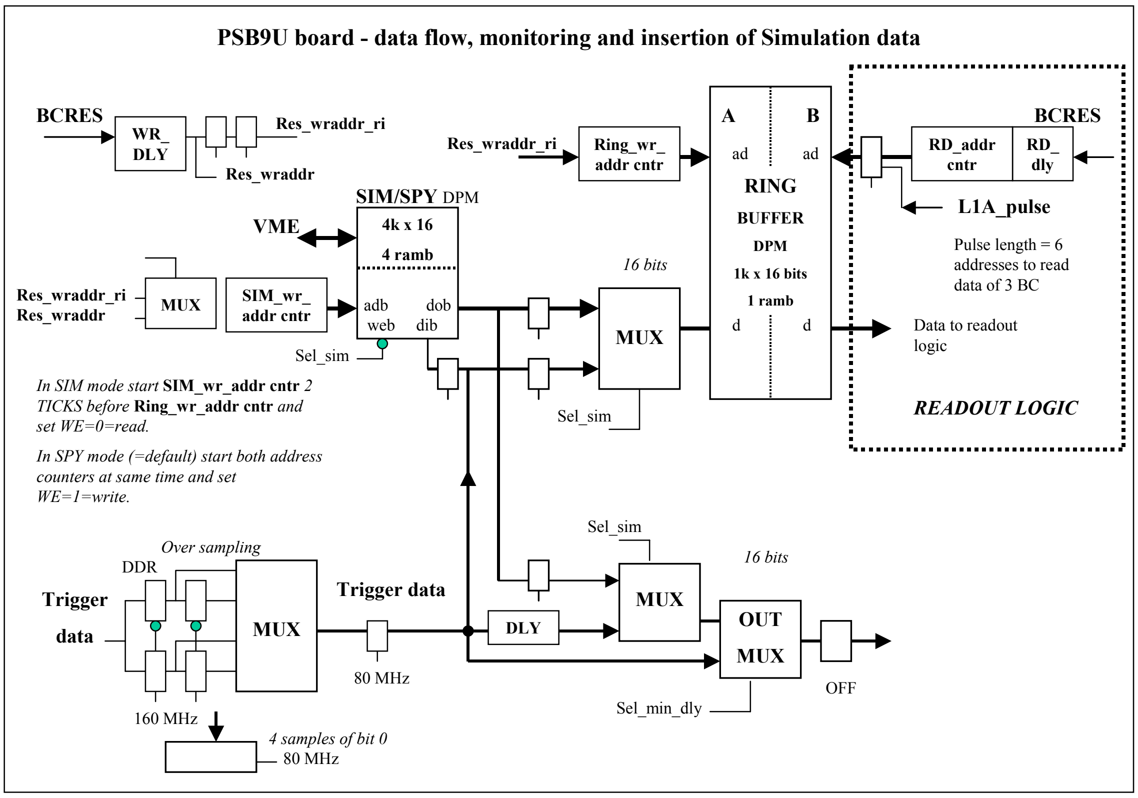


Figure 2 PSB9U data flow and monitoring scheme

For each 16-bit data stream a Ring Buffer memory monitors the trigger data. The memory runs in dual port mode. At one side the delayed trigger data are written continuously into the memory using addresses generated by a counter. The common Bunch Crossing Reset (BCRES) signal clears the counter synchronously to the clock thereby locking the addresses to the LHC orbit. After the end of the memory the write procedure continues at the first address overwriting old data. At the other side of the memory the DAQ-readout circuit applies a read address to get the data of the bunch crossing that has generated a L1A trigger signal. The read address is also supplied by a counter running synchronously to the LHC orbit. But a delayed BRES signal clears the read counter later than the write counter to compensate the trigger latency, that is the period between the writing time and a read access due to a L1A signal generated by trigger data of this bunch crossing. The length of the Ring Buffer memory (256 words) far exceeds the trigger latency.
Each (pair of) 16-bit channel(s) is locked separately to the LHC clock and to the orbit to compensate different cable delays.
The SYNC chip contains a second set of memories to load test or simulation data and to send them either once or repeatedly instead of trigger data to the Ring-Buffer and via the back-plane to the GTL respectively to the GMT board.
The readout circuits and a Derandomizing Buffer are either implemented in the SYNC chip or in a dedicated ROP chip if more than one SYNC chip per board is required.

1.1 Parallel LVDS input data

The PSB9U accepts also up to 64 bits of parallel trigger data alternatively to data from one Infiniband cable. The 40 MHz parallel input bits are accepted as LVDS signals received by 16 RJ45 connectors, each accepting 4 signals. The PSB board expects negative logic for differential signals. **Trigger bit =1 is expected with a negative voltage difference.** The SYNC chip inverts the values back to the true input levels before transmission.

1.1.1 Synchronization of 40 MHz parallel trigger data

As the precise arrival time of the data bits is unknown the SYNC chip first samples the input bits 4 times per bunch-crossing period (~25 ns) to find the switching point of the input data.
Phase selection and delay adjustment are done separately for each 4-bit group to consider time skew between cables and link chips. The SYNC chip takes the sample furthest away from the switching time, delays it for a programmable period, multiplexes the data into a 80 MHz data stream and sends the data as GTL+ signals over the back-plane to the logic board (GTL).
The SYNC chip also writes the input data into a RING BUFFER and in addition into a SPY memory for monitoring. The Ring Buffer runs continuously overwriting old data. If a Level-1 Accept (L1A) signal arrives at the PSB board, data are moved from the Ring Buffer into a Derandomizing Memory to be transferred later by the Readout Processor (ROP) to the readout board (GTFE).

1.1.1.1 Totem Trigger bits

Up to 16 bits are booked be the TOTEM Trigger (May 2004).

1.1.1.2 Free trigger bits (48)

48 bits are free (May 2004).

1.1.2 Synchronization of Technical Trigger bits

One PSB module accepts ‘Technical Trigger’ signals, which might not arrive as 40 MHz pulses. Therefore an edge sensing circuit generates synchronously to the system clock signal a 25ns pulse if either a rising or falling edge has been detected.

2 Keywords

2.1 RESET Signals

POWER OFF and ON
To switch the GT-crate off is the last option to reset non-working Global Trigger electronics.
NSYSRES ➔ configuration of FPGAs

The common crate-reset signal NSYSRES starts the configuration procedure for all FPGAs except the VME64 chip. It pulls the NPROG net to a low voltage level forcing the VME and the PSB chip to reconfigure from Proms.

RESET_DCM_PSB

The VME chip sends **RESET_DCM_PSB** to resynchronize the clock in the PSB chip to the board CLK.

RESET_PSB and INACTIVE ➔STARTUP of PSB chip

The VME chip sends **RESET_PSB** to reset the STARTUP module inside the PSB chip and the common INACTIVE signal enables the IO-pins to switch from high-Z to active mode.
RESET_PSB ➔ GSR pin of STARTUP ➔ set initial default values of registers
INACTIVE ➔ GTS pin of STARTUP
The RESET_PSB should reload the initial default values into all registers.

L1RES, BCRES, L1A ➔ reset State Machines and Counters

The Trigger Control System sends via the backplane the signals L1RES, BCRES, L1A and Event Counter Reset to reset state machines and counters

- L1RES clears the event data in the FIFOs.
- The ROC state machine returns always to IDLE state and cannot be reset. It stays in IDLE mode when the FIFOs are empty or when READY =0 (defined by software).

.

2.2 BCRES signal

It is possible that the distributed BCRES signal does not arrive every LHC orbit. An internal BC-counter and an Orbit_Length register are used to generate an internal BCRES signal running in phase with the distributed signal. Every circuit uses the internal BCRES signal to remain locked to the LHC orbit.

2.3 Sync check for BC0 data

Spy logic stores the arrival time of the SYNC bits as defined in the interface note CMS-IN-02-069.pdf for each 16 bit word. The number can be read by VME. A difference to a default value will be flagged as an error.

2.4 Phase check with over-sampling bits

All four samples of bit 0 are spied and compared to each other to find the time when the input data change their state. Four phase counters are incremented to check the stability of trigger data. At the end of every LHC orbit the counter contents are saved to be read by VME and the counters cleared. The contents are used to decide which sample should be taken as reliable data input.

2.5 Private Monitoring (option)

The 8kwords long SIM memory can run also in SPY mode to store a complete LHC orbit. In spy-mode the SIM/SPY memory will be written in parallel with the Ring-Buffer but can be read by VME. It can be used to get a ‘snap shot’ of the input data.

2.6 Test modes

- Read-back registers and memories in the PSB chip allow checking VME accesses.
- The SIM/SPY memory is used to send test- or simulation data to other boards.
- VME generated BCRES pulses and an on-board 40 MHz oscillator are used to run tests also in stand-alone mode.

2.7 SIM/SPY Memories and serial outputs

- For each channel a 8kW SIM/SPY memory can be used either to spy input data or to send simulation data to the back-plane.
- Channels 0…3 send the simulation data also to the transmitter part of their DS92LV16 Serial Link Chips. One output connector sends simulation data of channels 0 and 1, the other of channels 2 and 3.

2.8 Link test & bit error rate & reference memory

Input data can be compared against data from a Reference Memory that provides data either continuously or just during one LHC orbit.
Both the data source memory and the Reference memory have to be loaded with the same data set. The data source can be either in the Global Calorimeter electronics or on the same or a different PSB board.

The reference data are delayed to compensate the latency between the data source and the arriving time that is the time when input data are written into the SPY memory and the Ringbuffer.
Whenever the data words are different an error counter is incremented up to 'FFFF' where it stops. Reading the error counters also clears them.
This circuit is used to measure the bit error rate of the links with test data.

2.9 Cable loopback test

A cable is connected from a transmitter to a receiver connector.
Example: CH0,1 ➔ ➔ CH5,4
SIM/SPY memories 0 and 1 and the Reference memory are loaded with the same data set.
Channel 0 and 1 send Simulation data, maybe continuosly.
The Reference Memory runs also in the same mode as CH0 and 1.
Channel 4 and 5 receive data for only one orbit.
All error counters are read to clear them in advance.
The program sends now a 'start at next orbit' to all 4 channels and to the reference memory control.
After the transfer we can read the error counters and can compare the receiving spy memories against the reference data.

2.9.1 Bit error rate & cable loopback

With a setup as above but with all memories running continuously we read from time to time the error counters and write the results into a log file.

2.10 Configuration modes

2.10.1 PROM and JTAG

PROMS contain the default configuration that is loaded

- at start-up time or whenever
- a SYSRESET signal arrives from the VME or by a
- NPROG command sent by software.

The Proms are be loaded by JTAG.

- JTAG connectors are foreseen for the Parallel-CableIV interface that is used to download the configuration file from a Notebook-PC.
- JTAG via VME loads the PROMS via the emulated JTAG interface in the VME chip. A configuration program takes the configuration files and loads the data serially into the Proms.

The configuration options above require that the PSB chip has been set by solder-jumpers to MASTER MODE.

2.10.2 Other configuration options

For tests other configuration options are possible, but then the PSB chip has to be re-soldered to SLAVE mode. Using an optional modification on the PSB Mezzanine board it will/might also be possible to switch by a software command to Slave Mode.

2.11 Power

+5V and +3.3V are provided by the backplane. Linear voltage generators provide +2.5V and +1.5V for the FPGA chips.

2.12 Front Panel

240 mm = 4x60mm …4 4-fold Ethernet connectors
80 mm = 6 x 15 …6 Infiniband conn

7 mm LEMO connector
0mm = LEDs mounted above Infiniband connectors
= 327 mm

2.13 Sync IO-pins, ram-blocks

XC2V3000-4 BF957 mounted on MEZZ957 board
96 ramb; Mezz957: 641 io-pins connected to PSB9U board
565 IO-pins:
8x16 in+ 4x16 par_in + 4x16 out (DS92LV16; test)
+ (128+24)GTLp + 21 VREF+ 59 VME(32bit)
+ 29 ChLink + 16 RO-bus
+ 3(L1A,BCRES,L1RESET) +4 Status + 5Config + 20 VRN/VRP

82 RAM blocks:
8x4 Spy + 8 Ring + 8 Derand + (1Ring+1Derand for BCnr)

3 VME chip PSB - V1003

3.1 Versionshistory

- V1000: first design. Error at RESET_MODE. **DO NOT USE!!** (HB110805).
- V1001: based on V1000, but RESET_MODE error corrected. (HB110805).
- V1002: based on V1001, but PSB/MEM access (ENPSB, ENPSBMEM) made with DSSYNC for read and write accesses. (HB120805).
- **V1003:** based on V1002, but write/read for general-register and configuration-register implemented. (HB160805).

3.2 Logic description

The VME chip PSB works with the VME64x chip PSB as controller for the VME-bus of the PSB-9U-card. There are VME-registers on it as the Registers for Programmable-chips-configuration, General registers, Chip ID / version registers and JTAG registers. The VME-accesses to the PSB-chip are made via this chip too.

3.3 VME access

3.3.1 Base address

Base address of all GT-slaves is encoded on A31-A25 (A24 not used), because of address space of GTL-6U-card. See definition in VME64x-chip for PSB.

3.3.2 AM and datatransfer

AM=0x0D and 0x09 „extended data access“ - for single access.
AM=0x0F and 0x0B „extended block transfer“ - for block transfer access.
D16 „word access” - for all accesses.
See definitions in VME64x-chip for PSB.

3.4 Chip selection on PSB-9U-card

With the VME addresses A23-A20 the chip selection is done on the PSB-9U-card.

A23	A22	A21	A20	Chip-name
0	0	0	0	VME chip PSB
0	0	0	1	PSB-chip-registers
0	0	1	0	PSB-chip-memories

3.5 VME chip PSB register

Registeraddresses:			
A31..A24	A23..A20	A19..A05	A06..A01,(00)
8 bits		13bits	6bits
Base address	0000	XXXX	Registers

3.5.1 VME chip PSB address-table

The address-table lists the address-offset which has to be combined with the base-address of the card.

A23-A00 => Register-name

Register for Programmable-chips-configuration:

`0x000000`	=>	CMD_ENPROG-register (write/read)
`0x000002`	=>	CMD_NPROG-register (write/read)
`0x000004`	=>	CMD_INIT-register (write/read)
`0x000006`	=>	STAT_INIT-register (read)
`0x000008`	=>	STAT_DONE-register (read)
`0x00000A`	=>	Configuration register PSB-chip (write)

General pulse registers:

`0x000010`	=>	Command pulse register (write)
`0x000012`	=>	Status pulse register (read)

General registers:

`0x000014`	=>	Command register (write/read)
`0x000016`	=>	Status register (read)

Chip ID and version registers:

`0x000020`	=>	chip_id_register_3 (read)
`0x000022`	=>	chip_id_register_2 (read)
`0x000024`	=>	chip_id_register_1 (read)
`0x000026`	=>	chip_id_register_0 (read)
`0x000028`	=>	version_register_3 (read)
`0x00002A`	=>	version_register_2 (read)
`0x00002C`	=>	version_register_1 (read)
`0x00002E`	=>	version_register_0 (read)

JTAG registers:

`0x000030` =>	tdo_register (write)
`0x000032` =>	tdi_register (read)
`0x000034` =>	tms0_register (write)
`0x000036` =>	tms1_register (write)
`0x000038` =>	cnt32_register (write)
`0x00003A` =>	mode0_register (write/read)
`0x00003C` =>	mode1_register (write/read)
`0x00003E` =>	mode2_register (write/read)

Serial link mode registers:

`0x000040` =>	SERLINK0-register (write/read)
`0x000042` =>	SERLINK1-register (write/read)
`0x000044` =>	SERLINK2-register (write/read)
`0x000046` =>	LOCKED-register (read)

EN_TTIN Register to enable Technical or Totem Trigger bits:

`0x000050` =>	EN_TTIN -register (write/read)

Access to/from PSB-chip:

`0x1XXXXX` =>	see PSB-chip-registers
`0x2XXXXX` =>	see PSB-chip-memories

3.5.2 Register for Programmable-chips-configuration

The PSB-chip (Virtex-II) is configurable by configuration device and by VMEbus instructions. The selection is made by jumpers. The register-definition for configuration by VMEbus shall be a standard. See P:\Lab3Lib\Altera\Lab3_altera\sch\xilinx_conf.

Register names	**D7..D1**	**D0**
CMD_ENPROG	-	ENPROG_PSB
CMD_NPROG	-	NPROG_PSB
CMD_INIT	-	INIT_PSB
STAT_INIT	-	INIT_PSB
STAT_DONE	-	DONE_PSB

3.5.2.1 CMD_ENPROG-register

`0x000000` => CMD_ENPROG-register (write/read)

Bit 0 of the CMD_ENPROG-register allows sending the configuration bits via VME-bus to the PSB-chip.

3.5.2.2 CMD_NPROG-register

`0x000002` => CMD_NPROG-register (write/read)

Data-bit 0 = 1 of this register set the NPROG-signal of PSB-chip active. Then it should be reset to ‘0’. Then the PSB-chip enters into the configuration procedure. The FPGA either waits for configuration data (slave mode) sent via VME or starts to read configuration bits from a serial PROM (master mode).

3.5.2.3 CMD_INIT-register

`0x000004` => **CMD_INIT-register (write/read)**

Data-bit 0 = 1 of this register set the NINIT-signal of PSB-chip active.

3.5.2.4 STAT_INIT-register

`0x000006` => **STAT_INIT-register (read)**

Read the status of the NINIT-signal of PSB-chip (data-bit 0)

3.5.2.5 STAT_DONE-register

`0x000008` => **STAT_DONE-register (read)**

Read the status of the DONE-signal of PSB-chip (data-bit 0). After a successful configuration the PSB-chip sets DONE = 1.

3.5.2.6 Configuration register PSB-chip

`0x00000A` => **Configuration-register PSB-chip (write)**

The register is used to load the configuration bits into the PSB-chip (Virtex-II).
A write access to this register generates a CCLK and sends the data-bit 0 as DIN-signal to the PSB-chip, if the CMD_ENPROG-register bit has been set before. The VME accesses are repeated until the last bit has been loaded into the PSB-chip.

3.5.3 General pulse registers

Register names	D3	D2	D1	D0
Command_Pulse _Reg	SET_RUNNING (pulse)	RESET_PSB (pulse*)	RES_DCM_ PSB (pulse)	PWRDWN_ PSB (pulse)
Status_Pulse_Reg	RUNNING	LOCKED_ LED	CLK_LOCKED _PSB	not used

*) also generated by RESET_MODE

Register names	D7	D6	D5	D4
Command_Pulse _Reg	not used	not used	not used	not used
Status_Pulse_Reg	not used	not used	not used	not used

3.5.3.1 Command pulse register

`0x000010` => **Command-pulse-register (write)**

D0: **PWRDWN_PSB = 1** sends a low active pulse to the PSB-chip setting it into power down mode. NPWRDWN_B is sent as an open drain signal from the VME-PSB-chip to the PSB-chip.
Remark from data sheet:
The power-down sequence enables a designer to set the device into a low-power, inactive state. The sequence is initiated by pulling the PWRDWN_B pin Low. To monitor power-down status, observe the PWRDWN_B pin. When asserted, power-down has completed. After a successful wake-up, the status pin de-asserts. While powered down, the only active pins are the PWRDWN_B and DONE. All inputs are off and all outputs are 3-stated. While in the POWERDOWN state, the Power On Reset (POR) circuit is still active, but it does not reset the device if V_{CCINT}, V_{CCO}, or V_{CCAUX} falls below its minimum value. The POR circuit waits until the PWRDWN_B pin is released before resetting the device. Also, the PROG_B pin is not sampled while the device is in the POWERDOWN state. The PROG_B pin becomes active when the PWRDWN_B pin is released. Therefore, the device cannot be reset while in the

POWERDOWN state. The wake-up sequence is the reverse of the power-down sequence.

D1: **RES_DCM_PSB = 1** sends a high active pulse to the PSB-chip, to forces the DCM module to lock.

D2: **RESET_PSB = 1** sends a high active pulse to the PSB-chip for reset activities. (RESET_MODE is another source for RESET_PSB.)

D3: **SET_RUNNING = 1** sends a high active pulse to set board in RUNNING mode.

3.5.3.2 Status pulse register

`0x000012` => **Status-pulse-register (read)**

D0: not used.

D1: **CLK_LOCKED_PSB = 1** indicates, that the DCM module of the PSB chip are locked to the 40 MHz clock.
This status bit has to be checked immediately after the configuration of the PSB chip and before any other actions. If the chips do not lock then either the clock signal from the TIM board or the on-board oscillator are bad.

D2: **LOCKED_LED** is the status of an AND of all LOCKED-signals.
The LOCKED_LED signal will illuminate the front-panel LED only if all enabled Serial Receiver Chips are locked to the clock of incoming serial data and if the PSB Chip has locked to the system clock.

D3: **RUNNING = 1** board is active.
If RUNNING = 0, send a SET_RUNNING command via VME.

3.5.4 General registers

Register names	**D3**	**D2**	**D1**	**D0**
Command_Reg	V_SEL_ CABLES	VME_CONF	EN_ROBUS	EN_CHLINK
Status_Reg	not used	STATUS_SEL_ VME	not used	EN_CHLINK

Register names	**D7**	**D6**	**D5**	**D4**
Command_Reg	not used	not used	not used	V_SEL_ BACKPL
Status_Reg	not used	not used	JTAG_JUMPER	not used

3.5.4.1 Command register

`0x000014` => **Command-register (write)**

D0: **EN_CHLINK = 1** enables channel-link-chips, if CLK_LOCKED_PSB is active (signal NEN_CHLINK active).

D1: **EN_ROBUS = 1** enables ROBUS (signal NEN_ROBUS active).

D2: **VME_CONF = 1** enables configuration of PSB-chip via VME and switches external mux from PROM to VME.

D3: **V_SEL_CABLES = 1** switches JTAG-chains to cables (MasterBlaster and Parallel-Cable-IV).

D4: **V_SEL_BACKPL = 1** switches JTAG-chains to backplane connection via SCANPSC110 (if V_SEL_CABLES = 0).

Truthtable for D4 and D3:

D4	**D3**	
0	0	JTAG-chains via VME

X	1	JTAG-chains via cables
1	0	JTAG-chains via backplane

3.5.4.2 Status register

`0x000016` => Status-register (read)

D0: **EN_CHLINK** is the inverted status of signal NEN_CHLINK.
D1: not used.
D2: **STATUS_SEL_VME = 1** indicates, that configuration of PSB-chip via VME is selected.
For configuration of PSB-chip via VME, set VME_CONF = 1 in the Command-register.
D3: not used.
D4: not used.
D5: **JTAG_JUMPER = 1** indicates, that SEL_CABLE_JTAG-jumper (JP50) is inserted. Therefore JTAG-chains are connected to cables (MasterBlaster and Parallel-Cable-IV).
For changing the sources of JTAG-chains, remove the jumper and make the selection with V_SEL_CABLES and V_SEL_BACKPL in the Command-register.

3.5.5 Chip_ID and version registers

3.5.5.1 Definitions

Chip_id_register and version_register have fixed values in the hardware. These registers have read access only.
The versions `0x00000000 - 0x00000FFF` are used for tests.
The versions `0x00001000 - 0xFFFFFFFF` are used for runs in CMS.

3.5.5.2 Settings

PSB-9U-card Nr.1:

chip_id:	`0x00018121`
version:	`0x00001003`

3.5.5.3 Chip_ID and version registers addresses

`0x000020` => chip_id_register_3 (read)

D7	D6	D5	D4	D3	D2	D1	D0
chip_ID [31..24]							

`0x000022` => chip_id_register_2 (read)

D7	D6	D5	D4	D3	D2	D1	D0
chip_ID [23..16]							

`0x000024` => chip_id_register_1 (read)

D7	D6	D5	D4	D3	D2	D1	D0
chip_ID [15..08]							

`0x000026` => chip_id_register_0 (read)

D7	D6	D5	D4	D3	D2	D1	D0
chip_ID [07..00]							

`0x000028` => version_register_3 (read)

D7	D6	D5	D4	D3	D2	D1	D0
version [31..24]							

`0x00002A` => version_register_2 (read)

D7	D6	D5	D4	D3	D2	D1	D0
version [23..16]							

`0x00002C` => version_register_1 (read)

D7	D6	D5	D4	D3	D2	D1	D0
version [15..08]							

`0x00002E` => version_register_0 (read)

D7	D6	D5	D4	D3	D2	D1	D0
version [07..00]							

3.5.6 JTAG-registers

3.5.6.1 Definitions

JTAG registers are used to control JTAG-chains via VME-bus.
For details see JTAGController.vhd from Hannes Sakulin.

3.5.7 Serial link mode registers

Register names	**D15..D12**	**D11..D8**	**D7..D4**	**D3..D0**
SERLINK0	SEND_SYNC_ PATTERN [3..0]	LINE_ LOOPBACK [3..0]	TPWRDWN [3..0]	ENTR [3..0]
SERLINK1	RPWRDWN [7..0]		ENREC [7..0]	
SERLINK2	not used		LOCAL_LOOPBACK [7..0]	
LOCKED	not used		LOCKED [7..0]	

Remarks:

23.805 AT corrected in table above NTPWRDWN to TPWRDWN, NRPWRDWN to RPWRDWN.

Index [7..0] means DS92LV16 chip number = channel number.

SEND_SYNC_PATTERN = 1 => transmitter sends SYNC patterns so that a receiver can synchronize to the incoming data stream.
LINE_LOOPBACK = 1 => the serial received data are returned via the serial transmission line.
TPWRDWN = 1 => powers down the transmitter part of the chip (signal NTPWRDWN=0).
ENTR = 1 => enables the transmitter circuits of the chip.
LOCAL_LOOPBACK = 1 => returns parallel Transmit-data to parallel Receiver lines.
RPWRDWN = 1 => powers down the receiver part of the chip (signal NRPWRDWN=0).
ENREC = 1 => enables the receiver circuits of the DS92LV16 chip.

`0x000040` => **SERLINK0-register (write/read)**
`0x000042` => **SERLINK1-register (write/read)**
`0x000044` => **SERLINK2-register (write/read)**
`0x000046` => **LOCKED-register (read)**

3.5.8 EN_TTIN Register to enable Technical or Totem Trigger bits

Enables the LVDS receivers for Technical Trigger resp. Totem Trigger signals.
Disabled Receivers send bits 1111 = 'F'.
See also registers in PSB9U chip to switch between Parallel LVDS and Serial input channels.

Register names	**D15..D0**
EN_TTIN	EN_R[15..0]

EN_Rxx = 1 => enables parallel LVDS receivers for Parallel cable xx (0 = default).

`0x000050` => **EN_TTIN-register (write/read)**

3.6 DTACK/BERR-generation

Writing to writeable registers and reading from readable registers generates a DTACK signal.
Access to/from PSB-chip generates a DTACK signal.
No BERR signal is generated, always inactive!

4 PSB chip Adresses

4.1 Version history

4.1.1 V0012

VME dtack will be removed earlier, circuit against short pulse during en_psb removed
New **Test Signal**: **bc0_data_ch4** ….BC0 data are detected in Channel_4 (bit 15 =’1’ three times …01011101010…)

4.1.2 V0010, V0011

New: Reference Memory, 8 COMP_DLY registers, 16 Error Counters, new Testpoints
To measure the bit error rate of the Serial Links the module ‘traffic_police’ has been included. A Reference memory (8kx16 bit DPM) can be loaded with reference data to compare them with input data. Any difference increments error counters. See also the ‘Keywords’ chapter for a short description.
DTACK for read access: becomes active now 3 ticks after begin of ‘EN_PSB’
IOSTANDARD: GTLP for CHxx, LVDCI_DV2_33 for TRxx (data to DS92LV16)

4.1.3 V0009

This version is used for first longtime tests.
6.Sept 2005 = 14. implementation of psb_chip_struct
C:\GlobalTrigger\Psb\Psb_chip\psb_chip_lib\ps\psb_chip_struct\psb_chip_impl_14
chip_ID number =8131 chip_version =0009
*** New test points to check SIM/SPY mem 1
*** New : psb_chip_6Sept05.ucf ...one period for CLKIN
*** Timing: RECN3(15)= 9+3.07 > 12.0ns.....accepted by AT.
IOSTANDARD: GTLP for CHxx, LVDCI_DV2_33 for TRxx (data to DS92LV16)

4.2 Overview VME Addresses

A31-A24: = ‘BB’ = base address
A23-20: = 0001 …PSB chip
(A23-20: = 0002 ...Address space for other PSB chip memories is not used in present design)

19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	Register space (1kx16 readback)										
0	0	0	0	0	0	0	0	1	Read only status registers										
0	0	1	0	0	0	Sim_Spy_memory 0 (8kx16)													
0	0	1	0	0	1	Sim_Spy_memory 1 (8kx16)													
0	0	1	0	1	0	Sim_Spy_memory 2 (8kx16)													
0	0	1	0	1	1	Sim_Spy_memory 3 (8kx16)													
0	0	1	1	0	0	Sim_Spy_memory 4 (8kx16)													
0	0	1	1	0	1	Sim_Spy_memory 5 (8kx16)													
0	0	1	1	1	0	Sim_Spy_memory 6 (8kx16)													
0	0	1	1	1	1	Sim_Spy_memory 7 (8kx16)													
0	1	0	0	0	0	Reference_memory (8kx16)													

4.3 Sim Spy Memories

Remark: In present chip design the memories are also in same address space as the registers.

BB12 0000	SIM_SPY_MEM0	w/r
BB12 4000	SIM_SPY_MEM1	w/r
BB12 8000	SIM_SPY_MEM2	w/r
BB12 C000	SIM_SPY_MEM3	w/r
BB13 0000	SIM_SPY_MEM4	w/r
BB13 4000	SIM_SPY_MEM5	w/r
BB13 8000	SIM_SPY_MEM6	w/r
BB13 C000	SIM_SPY_MEM7	w/r
BB14 0000	REFERENCE_MEM	w/r

4.4 Register overview

BB10 0000	CHAN_REG0	w/r	
BB10 0002	CHAN_REG1	w/r	
BB10 0004	CHAN_REG2	w/r	
BB10 0006	CHAN_REG3	w/r	
BB10 0008	CHAN_REG4	w/r	
BB10 000A	CHAN_REG5	w/r	
BB10 000C	CHAN_REG6	w/r	
BB10 000E	CHAN_REG7	w/r	
***** Delay Registers for Serial Link channels*****			
BB10 0010	CHAN_DELAY0	w/r	
BB10 0012	CHAN_DELAY1	w/r	
BB10 0014	CHAN_DELAY2	w/r	
BB10 0016	CHAN_DELAY3	w/r	
BB10 0018	CHAN_DELAY4	w/r	
BB10 001A	CHAN_DELAY5	w/r	
BB10 001C	CHAN_DELAY6	w/r	
BB10 001E	CHAN_DELAY7	w/r	
***** Delay Registers for parallel LVDS data channels*****			
BB10 0020	LVDS_DELAY0	w/r	// bit 3-0
BB10 0022	LVDS_DELAY1	w/r	// bit 7-4
BB10 0024	LVDS_DELAY2	w/r	
BB10 0026	LVDS_DELAY3	w/r	
BB10 0028	LVDS_DELAY4	w/r	
BB10 002A	LVDS_DELAY5	w/r	
BB10 002C	LVDS_DELAY6	w/r	
BB10 002E	LVDS_DELAY7	w/r	
BB10 0030	LVDS_DELAY8	w/r	
BB10 0032	LVDS_DELAY9	w/r	
BB10 0034	LVDS_DELAY10	w/r	
BB10 0036	LVDS_DELAY11	w/r	
BB10 0038	LVDS_DELAY12	w/r	
BB10 003A	LVDS_DELAY13	w/r	
BB10 003C	LVDS_DELAY14	w/r	
BB10 003E	LVDS_DELAY15	w/r	// bit 63-60
***** Setup Registers (setup_reg)*****			
BB10 0040	BOARD_ID	w/r	
BB10 0042	BCRES_DELAY	w/r	

BB10 0044	LATENCY_DELAY	w/r	
BB10 0046	ROP_SETUP	w/r	
BB10 0048	MAX_BC_NUMBER	w/r	//=orbit length -1
BB10 004A	SEL_PHASE3100	w/r	// select phases for LVDS data
BB10 004C	SEL_PHASE6332	w/r	// select phases for LVDS data
BB10 004E	IDLE_ID_LOW	w/r	// idle identifier low part

(setup_reg1)

BB10 0050	IDLE_ID_HIGH	w/r	// idle identifier high part
BB10 0052	TESTMASK0	w/r	// tespoint 0 bits
BB10 0054	TESTMASK1	w/r	// tespoint 1 bits
BB10 0056	TESTMASK2	w/r	// tespoint 2 bits
BB10 0058	TESTMASK3	w/r	// tespoint 3 bits
BB10 005A	TESTMASK4	w/r	// tespoint 4 bits
BB10 005C	TESTMASK5	w/r	// tespoint 5 bits
BB10 005E	TESTMASK6	w/r	// tespoint 6 bits

***** Comparator Delays for data from Reference Memory *** (setup_reg2)**

BB10 0060	COMP_DLY0	w/r
BB10 0062	COMP_DLY1	w/r
BB10 0064	COMP_DLY2	w/r
BB10 0066	COMP_DLY3	w/r
BB10 0068	COMP_DLY4	w/r
BB10 006A	COMP_DLY5	w/r
BB10 006C	COMP_DLY6	w/r
BB10 006E	COMP_DLY7	w/r

***** Write Only Command Pulses (setup_reg3) ******

BB10 0070	CMD_PULSE	w/-
BB10 0072	REF_REG	w/r

++++++++++++++++++ READ ONLY ADDRESSES +++++++++++++++

***** Phase Counters for Serial Link channels ******

BB10 0800	PHASE_CNTR_A0	-/r	// compares ph1-ph0 and ph0-pre3
BB10 0802	PHASE_CNTR_ A1	-/r	
BB10 0804	PHASE_CNTR_A2	-/r	
BB10 0806	PHASE_CNTR_A3	-/r	
BB10 0808	PHASE_CNTR_A4	-/r	
BB10 080A	PHASE_CNTR_A5	-/r	
BB10 080C	PHASE_CNTR_A6	-/r	
BB10 080E	PHASE_CNTR_A7	-/r	
BB10 0810	PHASE_CNTR_B0	-/r	compares ph3-ph2 and ph2-1
BB10 0812	PHASE_CNTR_B1	-/r	
BB10 0814	PHASE_CNTR_B2	-/r	
BB10 0816	PHASE_CNTR_B3	-/r	
BB10 0818	PHASE_CNTR_B4	-/r	
BB10 081A	PHASE_CNTR_B5	-/r	
BB10 081C	PHASE_CNTR_B6	-/r	
BB10 081E	PHASE_CNTR_B7	-/r	

***** Phase Counters for parallel LVDS data channels ******

BB10 0820	PHASE_CNTR_A0_3	// compares ph1-ph0 and ph0-pre3 of bits 0-3
BB10 0822	PHASE_CNTR_A4_7	// compares ph1-ph0 and ph0-pre3 of bits 4-7
BB10 0824	PHASE_CNTR_A8_11	

BB10 0826	PHASE_CNTR_A12_15	
BB10 0828	PHASE_CNTR_A16_19	
BB10 082A	PHASE_CNTR_A20_23	
BB10 082C	PHASE_CNTR_A24_27	
BB10 082E	PHASE_CNTR_A28_31	
BB10 0830	PHASE_CNTR_A32_35	
BB10 0832	PHASE_CNTR_A36_39	
BB10 0834	PHASE_CNTR_A40_43	
BB10 0836	PHASE_CNTR_A44_47	
BB10 0838	PHASE_CNTR_A48_51	
BB10 083A	PHASE_CNTR_A52_55	
BB10 083C	PHASE_CNTR_A56_59	
BB10 083E	PHASE_CNTR_A60_63	// compares ph1-ph0 and ph0-pre3 of bits 60_63
BB10 0840	PHASE_CNTR_B0_3	// compares ph3-ph2 and ph2-1 of bits 0-3
BB10 0842	PHASE_CNTR_B4_7	// compares ph3-ph2 and ph2-1 of bits 4-7
BB10 0844	PHASE_CNTR_B8_11	
BB10 0846	PHASE_CNTR_B12_15	
BB10 0848	PHASE_CNTR_B16_19	
BB10 084A	PHASE_CNTR_B20_23	
BB10 084C	PHASE_CNTR_B24_27	
BB10 084E	PHASE_CNTR_B28_31	
BB10 0850	PHASE_CNTR_B32_35	
BB10 0852	PHASE_CNTR_B36_39	
BB10 0854	PHASE_CNTR_B40_43	
BB10 0856	PHASE_CNTR_B44_47	
BB10 0858	PHASE_CNTR_B48_51	
BB10 085A	PHASE_CNTR_B52_55	
BB10 085C	PHASE_CNTR_B56_59	
BB10 085E	PHASE_CNTR_B60_63	// compares ph3-ph2 and ph2-1 of bits 60_63

***** Status registers ******

BB10 0860	PSB_STATUS	-/r
BB10 0862	ROP_STATUS	-/r
BB10 0864	CHIP_ID	-/r
BB10 0866	VERSION_NR	-/r
BB10 0868	CHIP_IDH	-/r

***** Error Counters ******

BB10 0870	ERROR_COUNTER0	-/r	// REF to input of CH0
BB10 0872	ERROR_COUNTER1	-/r	
BB10 0874	ERROR_COUNTER2	-/r	
BB10 0876	ERROR_COUNTER3	-/r	
BB10 0878	ERROR_COUNTER4	-/r	
BB10 087A	ERROR_COUNTER5	-/r	
BB10 087C	ERROR_COUNTER6	-/r	
BB10 087E	ERROR_COUNTER7	-/r	// REF to input of CH7
BB10 0880	ERROR_COUNTER8	-/r	// REF to CH0_mem
BB10 0882	ERROR_COUNTER9	-/r	
BB10 0884	ERROR_COUNTER10	-/r	
BB10 0886	ERROR_COUNTER11	-/r	
BB10 0888	ERROR_COUNTER12	-/r	
BB10 088A	ERROR_COUNTER13	-/r	

```
BB10 088C     ERROR_COUNTER14                          -/r
BB10 088E     ERROR_COUNTER15                          -/r  // REF to CH7_mem
```

******************* **END OF OVERVIEW** **********************

4.5 Channel Registers

write & read		**functions**		
BB10 0000	CHAN_REG0	rx	tx	lvds
BB10 0002	CHAN_REG1	rx	tx	lvds
BB10 0004	CHAN_REG2	rx	tx	
BB10 0006	CHAN_REG3	rx	tx	
BB10 0008	CHAN_REG4	rx		
BB10 000A	CHAN_REG5	rx		
BB10 000C	CHAN_REG6	rx		
BB10 000E	CHAN_REG7	rx		

Channels 0,1: Parallel LVDS data can be received instead of the serial input data.
Channels 0,1,2,3: Simulation data can also be sent via the serial transmitter circuits to two front panel connectors.
Channels 4,5,6,7: receive serial data only.

CHAN_REG0 **bit 6: refmem_contin_mode**
= 1 The Reference Memory runs continously to check the incoming data
= 0 The Reference Memory runs for one orbit only after a '*start reference_mem at next orbit*' command pulse. See bit 15 of PSB_CMD_PULSE.

If we select simulation mode (**sel_sim_mode=1)** for these channels then LVDS data are not transferred to the FDL board (Technical trigger bits) and not to the GTL board (TOTEM trigger bits).

bits 15-6 : free
bit 5: **en_trx_data**
=1: send data from SIM_SPY memory also via DS92LV16 Serial Link transmitter to the Frontpanel connector.
bit 4: **sel_contin_mode**
=1: The SIM_SPY memory runs continuously either sending simulation data or storing input data
=0: The SIM_SPY memory runs for one orbit only and stops afterwards.
bit 3: **sel_sim_mode**
=1: The SIM_SPY memory sends simulation data to the backplane and if enabled in channels 0-3 also to the DS92LV16 Serial Link transmitters.
=0: The SIM_SPY memory stores input data for monitoring tasks.
For channels 0 and 1 the simulation data will replace either serial or parallel trigger data.
bit 2: **sel_lvdsdata**
…is valid for channels 0 and 1 only. For channels 2…7 it has to be set =0 otherwise no data will be transferred.
=1: Send parallel LVDS data to the backplane (Technical Trigger data)
=0: Send data from the DS92LV16 Serial Link to the backplane (Calorimeter trigger data)
bit 1: **sel_phase(1)**
bit 0: **sel_phase(0)**

Select Phase in over-sampling circuit to forward the trigger input data from the serial link. Take the sample that is most far from the data switching time.
V0005: Only phases 0 and 2 can be selected because of timing problems in the chip.
00 = take phase 0, **10** = take phase 2, **01** = inhibit data, **11** = inhibit data

4.6 Delay Registers for Serial Link Channels

write & read

BB10 0010	CHAN_DELAY0
BB10 0012	CHAN_DELAY1
BB10 0014	CHAN_ DELAY2
BB10 0016	CHAN_DELAY3
BB10 0018	CHAN_ DELAY4
BB10 001A	CHAN_DELAY5
BB10 001C	CHAN_ DELAY6
BB10 001E	CHAN_DELAY7

4.6.1 Programming Guideline for DELAYS

15 - 12	11 - 8	7 - 4	3 - 0	
Delay C	Delay B	Delay A	Delay= 0…3	

Total Delay = Delay C + Delay B + Delay A + (0…3)

```
  -- DELAY =0            →  0000 0000 0000 0000
  -- DELAY =1            → 0000 0000 0000 0001
  -- DELAY =2            → 0000 0000 0000 0010
  -- DELAY =3            → 0000 0000 0000 0011
  -- DELAY =C+B+A+3      → CCCC BBBB AAAA 0011
```

-- For DELAY <4 the bits 15-4 have to be =0 !!
-- Bits 3,2 are always =0; are not decoded

Remark about tests with different delays:

The SRL16 works like a shift register with an output multiplexer that can switch each shift register bit to the output as selected by A3,2,1,0.
If we change from a short to a long delay then it is possible that the shifted signal appears a second time at the output. Therefore we have to wait until the signal has been moved out before applying the new longer delay. However in real life the delay will not be changed during a run.

4.7 Delay Registers for parallel LVDS data channels

BB10 0020	LVDS_DELAY0	w/r	// bit 3-0
BB10 0022	LVDS_DELAY1	w/r	// bit 7-4
BB10 0024	LVDS_DELAY2	w/r	
BB10 0026	LVDS_DELAY3	w/r	
BB10 0028	LVDS_DELAY4	w/r	
BB10 002A	LVDS_DELAY5	w/r	
BB10 002C	LVDS_DELAY6	w/r	
BB10 002E	LVDS_DELAY7	w/r	
BB10 0030	LVDS_DELAY8	w/r	

BB10 0032	LVDS_DELAY9	w/r	
BB10 0034	LVDS_DELAY10	w/r	
BB10 0036	LVDS_DELAY11	w/r	
BB10 0038	LVDS_DELAY12	w/r	
BB10 003A	LVDS_DELAY13	w/r	
BB10 003C	LVDS_DELAY14	w/r	
BB10 003E	LVDS_DELAY15	w/r	// bit 63-60

4.7.1 Programming Guideline for DELAYS

15 - 12	11 - 8	7 - 4	3 - 0	
Delay C	Delay B	Delay A	Delay= 0…3	

Total Delay = Delay C + Delay B + Delay A + (0…3)

-- DELAY =0	➔ 0000 0000 0000 0000
-- DELAY =1	➔ 0000 0000 0000 0001
-- DELAY =2	➔ 0000 0000 0000 0010
-- DELAY =3	➔ 0000 0000 0000 0011
-- DELAY =C+B+A+3	➔ CCCC BBBB AAAA 0011

-- For DELAY <4 the bits 15-4 have to be =0 !!
-- Bits 3,2 are always =0; are not decoded

4.8 Board Identifier

BB10 0040 BOARD_ID **write & read**
16 bit word to identify the PSB board in the data record for CMS readout.

4.9 BCRES Delay

BB10 0042 BCRES_DELAY **write & read**
--

15 - 12	11 - 8	7 - 4	3 - 0	
Delay C	Delay B	Delay A	Delay= 0…3	

Total Delay = Delay C + Delay B + Delay A + (0…3)

-- DELAY =0	➔ 0000 0000 0000 0000
-- DELAY =1	➔ 0000 0000 0000 0001
-- DELAY =2	➔ 0000 0000 0000 0010
-- DELAY =3	➔ 0000 0000 0000 0011
-- DELAY =C+B+A+3	➔ CCCC BBBB AAAA 0011

-- For DELAY <4 the bits 15-4 have to be =0 !!
-- Bits 3,2 are always =0; are not decoded

4.10 Latency Delay

BB10 0044 LATENCY_DELAY **write & read**

15 - 12	11 - 8	7 - 4	3 - 0	

Delay C	Delay B	Delay A	Delay= 0…3	

Total Delay = Delay C + Delay B + Delay A + (0…3)

-- DELAY =0	➔ 0000 0000 0000 0000
-- DELAY =1	➔ 0000 0000 0000 0001
-- DELAY =2	➔ 0000 0000 0000 0010
-- DELAY =3	➔ 0000 0000 0000 0011
-- DELAY =C+B+A+3	➔ CCCC BBBB AAAA 0011

-- **For DELAY <4 the bits 15-4 have to be =0 !!**
-- Bits 3,2 are always =0; are not decoded

4.11 ROP Setup register

BB10 0046 ROP_SETUP **write & read**

Bit 15 – 4 are not used

Bit 3:**en_robus** =0 (default)

=1 enable the Bgo commands as the TIM board sends via the ROBUS

=0 the PSB uses the encoded command (L1Res, Bcres, L1A) signals sent by the TIM board.

Bit 2: **five_bx_event** =0 (default)

=1: A readout record contains data from 5 bunch crossings around the triggering bx (-2, -1, 0, +1, +2).

=0: A readout record contains data from 3 bunch crossings around the triggering bx (-1, 0, +1)

Bit 1 and bit 0: **PSB_MODE**

= 0 0 PSB is **DISCONNECTED** from readout (=default)

= 0 1 PSB is **BUSY** with other tasks and cannot receive any L1A for the time being. But the ROP, all counters and registers are correct to continue the data taking run.

= 1 0 PSB is **READY** and waits for the Bgo command ‘RUN’ to receive L1A and to send events to the GTFE board. For tests the ‘RUN’ command can also be simulated by a vme cmd pulse.

= 1 1 PSB sends **BAD CODE**…should never be set except for a test

4.12 MAX_BC_NUMBER

BB10 0048 MAX_BC_NUMBER w/r //=orbit length -1

Default value = 3563 dec = DEB hex

The number is used by a comparator to generate an internal bunch counter reset signal.

If it does not agree with external BCRES signal from the TIM board then a BC_ERROR flag will be set.

A BC_ERROR appears always with the first external BCRES and when sending a BCRes_vme signal. It has to be cleared by the command pulse ‘Res_BC_error’.

Remark: If MAX_BC_NUMBER = 0 then no BCRES will be generated inside the chip and the power consumption increases by 1-2 A.

4.13 SEL_PHASES for LVDS bits 63-00

BB10 004A	SEL_PHASE3100	w/r	// select phases for LVDS data
BB10 004C	SEL_PHASE6332	w/r	// select phases for LVDS data

SEL_PHASE3100	**15,14**	**13,12**	**11,10**	**9, 8**	**7, 6**	**5, 4**	**3, 2**	**1, 0**
Selects phases for LVDS bits:	31-28	27-24	23-20	19-16	15-12	11 - 8	7 - 4	3 - 0
SEL_PHASE6332	**15,14**	**13,12**	**11,10**	**9, 8**	**7, 6**	**5, 4**	**3, 2**	**1, 0**
Selects phases for LVDS bits:	63-60	59-56	55-52	51-48	47-44	43-40	39-36	35-32

00 ➔	selects phase sample 0
01 ➔	selects phase sample 1
10 ➔	selects phase sample 2
11 ➔	selects phase sample 3

4.14 Idle Identifier low

BB10 004E IDLE_IDL **write & read**

IDLE_IDL(15:0) defines the bits 15-0 that are sent between data records over the Channel Links to the GTFE readout board.

4.15 Idle Identifier high

BB10 0050 IDLE_IDH **write & read**

IDLE_IDH(11:0) defines the bits 27-16 that are sent between data records over the Channel Links to the GTFE readout board.

IDLE_IDH(15:12) =B"0000" are not used.

4.16 TESTPOINTS

BB10 0052	TESTMASK0	w/r	// tespoint 0 bits
BB10 0054	TESTMASK1	w/r	// tespoint 1 bits
BB10 0056	TESTMASK2	w/r	// tespoint 2 bits
BB10 0058	TESTMASK3	w/r	// tespoint 3 bits
BB10 005A	TESTMASK4	w/r	// tespoint 4 bits
BB10 005C	TESTMASK5	w/r	// tespoint 5 bits
BB10 005E	TESTMASK6	w/r	// tespoint 6 bits

TESTMASK0…6 select the signals that can be connected to the test points for monitored with an oscilloscope. If several signals per test point are selected then the signals are merged with an OR-function.

4.16.1 TESTMASKS for V0012

Red = new

bc0_data_ch4 = BC0 data in Channel 4 detected, signal appears 2 ticks later

bits	TESTMASK 0	TESTMASK 1	TESTMASK 2	TESTMASK 3
15	clk40	clk80	vme_wr	clr =/locked
14	bcres_int	bc_error	bcres_dlyed	bcres_dlyed1
13	res_evnr	res_orbitnr_i	l1res_int	run_rop
12	run_next_orbit(0)	rd_chan_reg(0)	wr_chan_reg(0)	chout0(15)
11	l1res_vme	**bc0_data_ch4**	l1a_int	l1a_int
10	psb_status(0)	psb_status(1)	psb_status(2)	psb_status(3)
9	start_rop_vme	stop_rop_vme	vme_rd_spy(0)	resevnr_vme
8	daq_data(24)	daq_data(25)	daq_data(26)	daq_data(27)
7	write_fifo	read_fifo	store_fifo_data	sclr_fifo
6	clr_ring_rdaddr	clr_ring_wradr	en_compp	inc_event_nr
5	sim_addr1(0)	sim_addr1(1)	sim_addr1(2)	sim_addr1(3)
4	sim_addr0(0)	sim_addr0(1)	sim_addr0(2)	sim_addr0(3)
3	vdout_ch1(0)	vdout_ch1(1)	vdout_ch1(2)	vdout_ch1(3)
2	inc_phas4(0)	inc_phas4(1)	inc_phas4(2)	inc_phas4(3)
1	inc_lvd0sph(0)	inc_lvd0sph(1)	inc_lvd0sph(2)	inc_lvd0sph(3)
0	stat_reg0(0)	stat_reg0(1)	stat_reg0(2)	stat_reg0(3)

bits	TESTMASK 4	TESTMASK 5	TESTMASK6
15	‘0’	vme_en	dtack
14	‘0’	we_spy_0	sim_mem0(0)
13	‘0’	vme_we_spy(0)	vme_en_spy(0)
12	‘0’	chout4(15)	trx0(15)
11	‘0’	bcres_vme	trx3(0)
10	‘0’	run_next_orbit_refmem	trx2(0)
9	‘0’	res_bc_error	trx1(0)
8	‘0’	run_next_orbit	trx0(0)
7	‘0’	chout7(0)	en_spy7
6	‘0’	chout6(0)	en_spy6
5	‘0’	chout5(0)	en_spy5
4	‘0’	chout4(0)	en_spy4
3	‘0’	chout3(0)	en_spy3

4.16.2 TESTMASKS for V0010

Red = new

en_compp = enables the comparators for Link tests

run_next_orbit_refmem = starts the Reference memory to send data in next orbit

bits	TESTMASK 0	TESTMASK 1	TESTMASK 2	TESTMASK 3
15	clk40	clk80	vme_wr	clr =/locked
14	bcres_int	bc_error	bcres_dlyed	bcres_dlyed1
13	res_evnr	res_orbitnr_i	l1res_int	run_rop
12	run_next_orbit(0)	rd_chan_reg(0)	wr_chan_reg(0)	chout0(15)
11	l1res_vme	l1a_int	l1a_int	l1a_int
10	psb_status(0)	psb_status(1)	psb_status(2)	psb_status(3)
9	start_rop_vme	stop_rop_vme	vme_rd_spy(0)	resevnr_vme
8	daq_data(24)	daq_data(25)	daq_data(26)	daq_data(27)
7	write_fifo	read_fifo	store_fifo_data	sclr_fifo
6	clr_ring_rdaddr	clr_ring_wradr	en_compp	inc_event_nr
5	sim_addr1(0)	sim_addr1(1)	sim_addr1(2)	sim_addr1(3)
4	sim_addr0(0)	sim_addr0(1)	sim_addr0(2)	sim_addr0(3)
3	vdout_ch1(0)	vdout_ch1(1)	vdout_ch1(2)	vdout_ch1(3)
2	inc_phas4(0)	inc_phas4(1)	inc_phas4(2)	inc_phas4(3)
1	inc_lvd0sph(0)	inc_lvd0sph(1)	inc_lvd0sph(2)	inc_lvd0sph(3)
0	stat_reg0(0)	stat_reg0(1)	stat_reg0(2)	stat_reg0(3)

bits	TESTMASK 4	TESTMASK 5	TESTMASK6
15	‘0’	vme_en	dtack
14	‘0’	we_spy_0	sim_mem0(0)
13	‘0’	vme_we_spy(0)	vme_en_spy(0)
12	‘0’	chout4(15)	trx0(15)
11	‘0’	bcres_vme	trx3(0)
10	‘0’	run_next_orbit_refmem	trx2(0)
9	‘0’	res_bc_error	trx1(0)
8	‘0’	run_next_orbit	trx0(0)
7	‘0’	chout7(0)	en_spy7
6	‘0’	chout6(0)	en_spy6
5	‘0’	chout5(0)	en_spy5
4	‘0’	chout4(0)	en_spy4
3	‘0’	chout3(0)	en_spy3
2	‘0’	chout2(0)	en_spy2
1	‘0’	chout1(0)	en_spy1
0	‘0’	chout0(0)	en_spy0

4.16.3 TESTMASKS for V0009

Red = new

bits	TESTMASK 0	TESTMASK 1	TESTMASK 2	TESTMASK 3
15	clk40	clk80	vme_wr	clr =/locked
14	bcres_int	bc_error	bcres_dlyed	bcres_dlyed1
13	res_evnr	res_orbitnr_i	l1res_int	run_rop
12	run_next_orbit(0)	rd_chan_reg(0)	wr_chan_reg(0)	chout0(15)
11	l1res_vme	l1a_int	l1a_int	l1a_int

10	psb_status(0)	psb_status(1)	psb_status(2)	psb_status(3)
9	start_rop_vme	stop_rop_vme	vme_rd_spy(0)	resevnr_vme
8	daq_data(24)	daq_data(25)	daq_data(26)	daq_data(27)
7	write_fifo	read_fifo	store_fifo_data	sclr_fifo
6	clr_ring_rdaddr	clr_ring_wradr	w_cmd_puls	inc_event_nr
5	sim_addr1(0)	sim_addr1(1)	sim_addr1(2)	sim_addr1(3)
4	sim_addr0(0)	sim_addr0(1)	sim_addr0(2)	sim_addr0(3)
3	vdout_ch1(0)	vdout_ch1(1)	vdout_ch1(2)	vdout_ch1(3)
2	inc_phas4(0)	inc_phas4(1)	inc_phas4(2)	inc_phas4(3)
1	inc_lvd0sph(0)	inc_lvd0sph(1)	inc_lvd0sph(2)	inc_lvd0sph(3)
0	stat_reg0(0)	stat_reg0(1)	stat_reg0(2)	stat_reg0(3)

bits	TESTMASK 4	TESTMASK 5	TESTMASK6
15	‘0’	vme_en	dtack
14	‘0’	we_spy_0	sim_mem0(0)
13	‘0’	vme_we_spy(0)	vme_en_spy(0)
12	‘0’	chout4(15)	trx0(15)
11	‘0’	bcres_vme	trx3(0)
10	‘0’	reset_error_flag	trx2(0)
9	‘0’	res_bc_error	trx1(0)
8	‘0’	run_next_orbit	trx0(0)
7	‘0’	chout7(0)	en_spy7
6	‘0’	chout6(0)	en_spy6
5	‘0’	chout5(0)	en_spy5
4	‘0’	chout4(0)	en_spy4
3	‘0’	chout3(0)	en_spy3
2	‘0’	chout2(0)	en_spy2
1	‘0’	chout1(0)	en_spy1
0	‘0’	chout0(0)	en_spy0

4.16.4 TESTMASKS for V0007 & V0008

bits	TESTMASK 0	TESTMASK 1	TESTMASK 2	TESTMASK 3
15	clk40	clk80	vme_wr	clr =/locked
14	bcres_int	bc_error	bcres_dlyed	bcres_dlyed1
13	res_evnr	res_orbitnr_i	l1res_int	run_rop
12	run_next_orbit(0)	rd_chan_reg(0)	wr_chan_reg(0)	chout0(15)
11	l1res_vme	l1a_int	l1a_int	l1a_int
10	psb_status(0)	psb_status(1)	psb_status(2)	psb_status(3)
9	start_rop_vme	stop_rop_vme	vme_rd_spy(0)	resevnr_vme
8	daq_data(24)	daq_data(25)	daq_data(26)	daq_data(27)
7	write_fifo	read_fifo	store_fifo_data	sclr_fifo
6	clr_ring_rdaddr	clr_ring_wradr	w_cmd_puls	inc_event_nr
5	rop_status(11)	rop_status(12)	rop_status(14)	rop_status(15)
4	sim_addr0(0)	sim_addr0(1)	sim_addr0(2)	sim_addr0(3)
3	inc_phas5(0)	inc_phas5(1)	inc_phas5(2)	inc_phas5(3)
2	inc_phas4(0)	inc_phas4(1)	inc_phas4(2)	inc_phas4(3)
1	inc_lvd0sph(0)	inc_lvd0sph(1)	inc_lvd0sph(2)	inc_lvd0sph(3)
0	stat_reg0(0)	stat_reg0(1)	stat_reg0(2)	stat_reg0(3)

bits	TESTMASK 4	TESTMASK 5	TESTMASK6
15	‘0’	vme_en	dtack
14	‘0’	we_spy_0	sim_mem0(0)
13	‘0’	vme_we_spy(0)	vme_en_spy(0)
12	‘0’	chout4(15)	trx0(15)
11	‘0’	bcres_vme	trx3(0)
10	‘0’	reset_error_flag	trx2(0)
9	‘0’	res_bc_error	trx1(0)
8	‘0’	run_next_orbit	trx0(0)
7	‘0’	chout7(0)	en_spy7
6	‘0’	chout6(0)	en_spy6
5	‘0’	chout5(0)	en_spy5
4	‘0’	chout4(0)	en_spy4
3	‘0’	chout3(0)	en_spy3
2	‘0’	chout2(0)	en_spy2
1	‘0’	chout1(0)	en_spy1
0	‘0’	chout0(0)	en_spy0

4.16.5 TESTMASKS for V0005, V0006:

bits	TESTMASK 0	TESTMASK 1	TESTMASK 2	TESTMASK 3
15	Clk40	Clk80	‘0’	clr =/locked
14	BCRes_int	bc_error	bcres_dlyed	bcres_dlyed1
13	Res_Evnr	Res_Orbitnr_i	L1Res_int	run_rop
12	Run_next_orbit(0)	en_spy_0	we_spy_0	chout0(15)
11	L1Res_vme	L1A_int	L1A_int	L1A_int
10	psb_status(0)	psb_status(1)	psb_status(2)	psb_status(3)
9	‘0’	‘0’	vme_wr	vme_en_spy
8	daq_data(24)	daq_data(25)	daq_data(26)	daq_data(27)
7	write_fifo	read_fifo	store_fifo_data	sclr_fifo
6	clr_ring_rdaddr	clr_ring_wradr	‘0’	inc_event_nr
5	rop_status(11)	rop_status(12)	rop_status(14)	rop_status(15)
4	rop_status(7)	rop_status(8)	rop_status(9)	rop_status(10)
3	inc_phas4(3)	inc_phas5(3)	inc_lvd0sph(3)	stat_reg0(3)
2	inc_phas4(2)	inc_phas5(2)	inc_lvd0sph(2)	stat_reg0(2)
1	inc_phas4(1)	inc_phas5(1)	inc_lvd0sph(1)	stat_reg0(1)
0	inc_phas4(0)	inc_phas5(0)	inc_lvd0sph(0)	stat_reg0(0)

bits	TESTMASK 4	TESTMASK 5	TESTMASK6
15	‘0’	vme_en	vme_en
14	‘0’	dtack	vme_wr
13	‘0’	vme_we_spy(0)	vme_rd_spy(0)
12	‘0’	rd_chan_reg(0)	wr_chan_reg(0)
11	‘0’	rd_chan_delay(0)	wr_chan_delay(0)
10	‘0’	rd_lvds_delay(0)	wr_lvds_delay(0)
9	‘0’	rd_setup_reg(0)	wr_setup_reg(0)
8	‘0’	rd_setup_reg1(0)	wr_setup_reg1(0)
7	‘0’	rd_stat_regs(0)	w_cmd_pulse
6	‘0’	rd_phase_cntb (0)	rd_phase_cnta (0)
5	‘0’	rd_phase_a6332 (0)	rd_phase_a3100 (0)
4	‘0’	rd_phase_b6332 (0)	rd_phase_b3100 (0)
3	‘0’	Start_rop_vme	Stop_rop_vme
2	‘0’	ResOrbnr_vme	reset_error_flag
1	‘0’	ResEvnr_vme	res_bc_error
0	‘0’	BCRes_vme	run_next_orbit

See ROP_STATUS bits: 15= roc_is_idle, 14=run_rop, 13=0, 12=out_of_sync, 11=error, 10=warning, 9=full_fifo, 8-0 = empty fifos.
ROP internal signals:

write_fifo
read_fifo, sclr_fifo
store_fifo_data
inc_event_nr
clr_ring_rdaddr, clr_ring_wradr

4.17 Comparator Delay Registers

write & read

BB10 0060	COMP_DLY0
BB10 0062	COMP_DLY1
BB10 0064	COMP_DLY2
BB10 0066	COMP_DLY3
BB10 0068	COMP_DLY4
BB10 006A	COMP_DLY5
BB10 006C	COMP_DLY6
BB10 006E	COMP_DLY7

The Delay register are used to delay data from the Reference Memory so that data from the same address of a transmitting memory are compared to each other.
The transmitting memory can be either

- another SIM memory on the same board sending their data via a cable back to another channel. ➔ CABLE LOOPBACK TEST
  - delay= depends from cable length (*=8x0.5bx for a 50cm cable)*
- another SIM memory on a different board ➔ LINK TEST between 2 PSB boards
- a memory in the GCT ➔ LINK TEST GCT to GT
  - delay = GCT_GT latency

See also ERROR COUNTERS below.

4.17.1 Programming Guideline for DELAYS

15 - 12	11 - 8	7 - 4	3 - 0	
Delay C	Delay B	Delay A	Delay= 0…3	

Total Delay = Delay C + Delay B + Delay A + (0…3)

```
-- DELAY =0          ➔ 0000 0000 0000 0000
-- DELAY =1          ➔ 0000 0000 0000 0001
-- DELAY =2          ➔ 0000 0000 0000 0010
-- DELAY =3          ➔ 0000 0000 0000 0011
-- DELAY =C+B+A+3    ➔ CCCC BBBB AAAA 0011
```

-- For DELAY <4 the bits 15-4 have to be =0 !!
-- Bits 3,2 are always =0; are not decoded

4.18 Command Pulses

BB10 0070 PSB_CMD_PULSE **write only**
A data bit =1 generates a spurious pulse to start any action in the PSB chip.

Bit 15: start reference_mem at next orbit	//Start Reference memory at begin of next orbit
Bit14: stop_rop_vme	// stop ROP state machine = 'stop run'
Bit13: start_rop_vme	// start ROP state machine making and sending events
Bit12: Res_BC_error	// reset BC error after startup and after BCRes_vme
Bit 11: Res_Evnr_vme	// reset Event Number Counter per software
Bit 10: Res_Orbitnr_vme	// reset Orbit Number Counter per software (not used)
Bit 9: BCRes_vme	// BCRES per software (for test only)
Bit 8: L1Res_vme	// simulate a L1Res (Resync) pulse

// in chip design: 'run_next_orbit(7:0)' = start sim_spy7…0 at next orbit

Bit 7: start sim_spy7 at next orbit	//Start Sim_Spy memory at begin of next orbit
Bit 6: start sim_spy6 at next orbit	
Bit 5: start sim_spy5 at next orbit	
Bit 4: start sim_spy4 at next orbit	
Bit 3: start sim_spy3 at next orbit	
Bit 2: start sim_spy2 at next orbit	
Bit 1: start sim_spy1 at next orbit	
Bit 0: start sim_spy0 at next orbit	

4.19 Phase Counters for Serial data

read only 32 8bit-counters

BB10 0800	PHASE_CNTR_A0	// compares ph1-ph0 and ph0-pre3 of chann 0
BB10 0802	PHASE_CNTR_ A1	// compares ph1-ph0 and ph0-pre3 of chann 1
BB10 0804	PHASE_CNTR_A2	
BB10 0806	PHASE_CNTR_A3	
BB10 0808	PHASE_CNTR_A4	
BB10 080A	PHASE_CNTR_A5	
BB10 080C	PHASE_CNTR_A6	
BB10 080E	PHASE_CNTR_A7	
BB10 0810	PHASE_CNTR_B0	// compares ph3-ph2 and ph2-1 of chann 0
BB10 0812	PHASE_CNTR_B1	// compares ph3-ph2 and ph2-1 of chann 1
BB10 0814	PHASE_CNTR_B2	
BB10 0816	PHASE_CNTR_B3	
BB10 0818	PHASE_CNTR_B4	
BB10 081A	PHASE_CNTR_B5	
BB10 081C	PHASE_CNTR_B6	
BB10 081E	PHASE_CNTR_B7	// compares ph3-ph2 and ph2-1 of chann 7

15 - 8	7 - 0	
Phase Counter 10	Phase Counter 0p3	PHASE_CNTR_Ax
Phase Counter 32	Phase Counter 21	PHASE_CNTR_Bx

x = 0…7 = Channel number

If the incoming data bit switches between two consecutive samples then a 8 bit Phase Counter will be incremented. If a phase counter becomes 'FF' then counting stops, showing an overflow. Reading of phase counters also clears their content.
Phase Counter A checks between sample pre3 and 0.
Phase Counter B checks between sample 0 and 1.
Phase Counter C checks between sample 1 and 2.
Phase Counter D checks between sample 2 and 3.
/pre3 = sample 3 of preceding 12.5 ns tick

4.20 Phase Counters for parallel LVDS data

read only 32 words for 64 8bit-counters

BB10 0820	PHASE_CNTR_A0_3	// compares ph1-ph0 and ph0-pre3 of bits 0-3
BB10 0822	PHASE_CNTR_A4_7	// compares ph1-ph0 and ph0-pre3 of bits 4-7
BB10 0824	PHASE_CNTR_A8_11	
BB10 0826	PHASE_CNTR_A12_15	
BB10 0828	PHASE_CNTR_A16_19	
BB10 082A	PHASE_CNTR_A20_23	
BB10 082C	PHASE_CNTR_A24_27	
BB10 082E	PHASE_CNTR_A28_31	
BB10 0830	PHASE_CNTR_A32_35	
BB10 0832	PHASE_CNTR_A36_39	
BB10 0834	PHASE_CNTR_A40_43	
BB10 0836	PHASE_CNTR_A44_47	
BB10 0838	PHASE_CNTR_A48_51	
BB10 083A	PHASE_CNTR_A52_55	
BB10 083C	PHASE_CNTR_A56_59	
BB10 083E	PHASE_CNTR_A60_63	// compares ph1-ph0 and ph0-pre3 of bits 60_63
BB10 0840	PHASE_CNTR_B0_3	// compares ph3-ph2 and ph2-1 of bits 0-3
BB10 0842	PHASE_CNTR_B4_7	// compares ph3-ph2 and ph2-1 of bits 4-7
BB10 0844	PHASE_CNTR_B8_11	
BB10 0846	PHASE_CNTR_B12_15	
BB10 0848	PHASE_CNTR_B16_19	
BB10 084A	PHASE_CNTR_B20_23	
BB10 084C	PHASE_CNTR_B24_27	
BB10 084E	PHASE_CNTR_B28_31	
BB10 0850	PHASE_CNTR_B32_35	
BB10 0852	PHASE_CNTR_B36_39	
BB10 0854	PHASE_CNTR_B40_43	
BB10 0856	PHASE_CNTR_B44_47	
BB10 0858	PHASE_CNTR_B48_51	
BB10 085A	PHASE_CNTR_B52_55	
BB10 085C	PHASE_CNTR_B56_59	
BB10 085E	PHASE_CNTR_B60_63	// compares ph3-ph2 and ph2-1 of bits 60_63

4.21 PSB Status register

BB10 0860 PSB_STATUS **read only**
Bit 15- 5 unused
Bit 4: BC_error

- =1 if the external BCRES signal and the BC-counter disagree. The length of the orbit is defined by the MAX_BC_NUMBER content.
- BC_error appears always after the initial power-up or DCM(clock) reset, the first external BCRES or after a BCRES_vme command.
- If BC_error becomes =1 during normal run then there are serious hardware problems, due to instable electronics.
- It has to be cleared by the **command pulse 'Res_BC_error'** before starting a run.
- The **'Res_BC_error'** pulse has to be sent at least *1 orbit after having loaded a new value* into the MAX_BC_NUMBER register.

Bit 3 -0 : encoded 4 bit status sent via FDL to the TCS Trigger Control board.

Bit3 Ready	Bit2 Busy	Bit1 Out_of_Sync	Bit0 Warning	Status of PSB
0	0	0	0	Disconnected
0	0	0	1	Warning
0	0	1	0	Out_of_Sync error
0	0	1	1	----
0	1	0	0	Busy
0	1	0	1	---
0	1	1	0	---
0	1	1	1	---
1	0	0	0	Ready
1	0	0	1	---
1	0	1	0	---
1	0	1	1	---
1	1	0	0	Error (not used by PSB)
1	1	0	1	---
1	1	1	0	---
1	1	1	1	Disconnected

Encoded Status of PSB board

The table agrees with the TCS Note.

4.22 ROP Status register

BB10 0862 ROP_STATUS **read only**

Bit 15: roc_is_idle	// The ROC Readout Controller state machine is in idle mode
Bit 14: run_flag	// 1= ROC is running when software sets the
	// ROP SETUP register = B"….10" = PSB READY=1
	// 0= either software or Bgo command has stopped
	// the ROC readout controller to extract and send events
Bit 13: 0	
Bit 12: out_of_sync	// empty bits of readout FIFOs did not appear at same time
Bit 11: error	// currently not implemented
Bit 10: warning	// more than 75% of the readout FIFO has been filled
Bit 9: full_fifo	// readout FIFOs are full ➔ error!!
Bit 8: empty(8)	// empty bit of readout FIFO 8 (BC number)
Bit 7: empty(7)	// empty bit of readout FIFO for channel 7
Bit 6: empty(6)	
Bit 5: empty(5)	
Bit 4: empty(4)	

Bit 3: empty(3)
Bit 2: empty(2)
Bit 1: empty(1)
Bit 0: empty(0)   // empty bit of readout FIFO for channel 0

4.23 CHIP Identifier

BB10 0864   CHIP_ID   **read only**
The 16 bit identifier is defined in the VHDL code for the PSB chip and cannot be changed by software.
CHIPID = 8131   8= PSB board, 1=cardnr, 3 = PSB chip, 1=chipnr (only one psb chip per board)

4.24 Version Number

BB10 0866   VERSION_NR   **read only**
The 16 bit identifier is defined in the VHDL code for the PSB chip and cannot be changed by software.
VERSION_NR = 0001….and higher

4.25 CHIP Identifier H

BB10 0868   CHIP_IDH   **read only**
The 16 bit identifier is defined in the VHDL code for the PSB chip and cannot be changed by software.
CHIPIDH = 0001   1= Global Trigger crate

4.26 ERROR COUNTERS

BB10 0870	ERROR_COUNTER0	-/r	// REF to input of CH0
BB10 0872	ERROR_COUNTER1	-/r	
BB10 0874	ERROR_COUNTER2	-/r	
BB10 0876	ERROR_COUNTER3	-/r	
BB10 0878	ERROR_COUNTER4	-/r	
BB10 087A	ERROR_COUNTER5	-/r	
BB10 087C	ERROR_COUNTER6	-/r	
BB10 087E	ERROR_COUNTER7	-/r	// REF to input of CH7
BB10 0880	ERROR_COUNTER8	-/r	// REF to CH0_mem
BB10 0882	ERROR_COUNTER9	-/r	
BB10 0884	ERROR_COUNTER10	-/r	
BB10 0886	ERROR_COUNTER11	-/r	
BB10 0888	ERROR_COUNTER12	-/r	
BB10 088A	ERROR_COUNTER13	-/r	
BB10 088C	ERROR_COUNTER14	-/r	
BB10 088E	ERROR_COUNTER15	-/r	// REF to CH7_mem

- Reading the Error Counters also clears the counters. Therefore before starting any tests all counters should be read.
- The value “FFFF” shows a counter overflow since the last read access.

ERROR_COUNTER0…7 shows any difference between input data into this channel and the reference data. The reference data are delayed by setting the COMP_DLY0…7 register so that the input and reference data of the same bunch crossing will be compared. The common BCRES signal is used to synchronize the data source electronics and the receiving PSB channel to each other.

ERROR_COUNTER8…15 shows any difference between the sending SIM0…7 memory and the reference data.

Example: CH0 sends data from the SIM memory to backplane or/and to the transmitting part of its serial link chip. Error Counter8 checks then if SIM0 and REF data agree.

5 PSB logic functions

5.1 Data Format of Channel Link

The table is defined for 5 bx per event. For normal events the record contains the parts for bx-1, bx+0, bx+1 only.
Normal record length=24 x 3 +1 = 73
Debug record length = 24 x 5 +1 = 121
Transfer time: 73 x 25 ns = 1800 ns resp. 121x 25= 3025 ns per event for 40 MHz Channel Link.

27-24	23-20	19-16	15-12	11-8	7-4	3-0	Name	Comment	Example
I	I	I	I	I	I	I	IDLE	Between records	555AAAA
A	0	0	e	e	e	e	HEADER A	EVNr(15:0)	A000001
B	0	0	0	0	e	e	HEADER B	EVNr(23:16)	0000000
C	0	0	bx-2	b	b	b	HEADER C	Bx_in_ev/bx of fifo	C00E017
D	0	0	n	n	n	n	HEADER D	Board identifier	D00ABCD
1	0	0	d	d	d	d	A_data ch0 of bx-2		
1	0	0	d	d	d	d	A_data ch1 of bx-2		
1	0	0	d	d	d	d	A_data ch2 of bx-2		
1	0	0	d	d	d	d	A_data ch3 of bx-2		
1	0	0	d	d	d	d	A_data ch4 of bx-2		
1	0	0	d	d	d	d	A_data ch5 of bx-2		
1	0	0	d	d	d	d	A_data ch6 of bx-2		
1	0	0	d	d	d	d	A_data ch7 of bx-2		
1	0	0	d	d	d	d	B_data ch0 of bx-2		
1	0	0	d	d	d	d	B_data ch1 of bx-2		
1	0	0	d	d	d	d	B_data ch2 of bx-2		
1	0	0	d	d	d	d	B_data ch3 of bx-2		
1	0	0	d	d	d	d	B_data ch4 of bx-2		
1	0	0	d	d	d	d	B_data ch5 of bx-2		
1	0	0	d	d	d	d	B_data ch6 of bx-2		
1	0	0	d	d	d	d	B_data ch7 of bx-2		
E	0	0	*000b*	b	b	b	End of bx-2	Ring addr of B_data	E000018
E	0	0	0	0	0	0	End of bx-2		E000000
E	0	0	0	0	0	0	End of bx-2		E000000
E	0	0	0	0	0	0	End of bx-2		E000000
A	0	0	e	e	e	e	HEADER A	EVNr(15:0)	A000001
B	0	0	0	0	e	e	HEADER B	EVNr(23:16)	0000000
C	0	0	bx-1	b	b	b	HEADER C	Bx_in_ev/bx of fifo	C00F019
D	0	0	n	n	n	n	HEADER D	Board identifier	D00ABCD
1	0	0	d	d	d	d	A_data ch0 of bx-1		
1	0	0	d	d	d	d	A_data ch1 of bx-1		
1	0	0	d	d	d	d	A_data ch2 of bx-1		
1	0	0	d	d	d	d	A_data ch3 of bx-1		
1	0	0	d	d	d	d	A_data ch4 of bx-1		
1	0	0	d	d	d	d	A_data ch5 of bx-1		
1	0	0	d	d	d	d	A_data ch6 of bx-1		
1	0	0	d	d	d	d	A_data ch7 of bx-1		
1	0	0	d	d	d	d	B_data ch0 of bx-1		
1	0	0	d	d	d	d	B_data ch1 of bx-1		

1	0	0	d	d	d	d	B_data ch2 of bx-1		
1	0	0	d	d	d	d	B_data ch3 of bx-1		
1	0	0	d	d	d	d	B_data ch4 of bx-1		
1	0	0	d	d	d	d	B_data ch5 of bx-1		
1	0	0	d	d	d	d	B_data ch6 of bx-1		
1	0	0	d	d	d	d	B_data ch7 of bx-1		
E	0	0	*000b*	b	b	b	End of bx-1	Ring addr of B_data	E00001A
E	0	0	0	0	0	0	End of bx-1		E000000
E	0	0	0	0	0	0	End of bx-1		E000000
E	0	0	0	0	0	0	End of bx-1		E000000
A	0	0	e	e	e	e	HEADER A	EVNr(15:0)	A000001
B	0	0	0	0	e	e	HEADER B	EVNr(23:16)	0000000
C	0	0	bx+0	b	b	b	HEADER C	Bx_in_ev/bx of fifo	C00001B
D	0	0	n	n	n	n	HEADER D	Board identifier	D00ABCD
1	0	0	d	d	d	d	A_data ch0 of bx+0		
1	0	0	d	d	d	d	A_data ch1 of bx+0		
1	0	0	d	d	d	d	A_data ch2 of bx+0		
1	0	0	d	d	d	d	A_data ch3 of bx+0		
1	0	0	d	d	d	d	A_data ch4 of bx+0		
1	0	0	d	d	d	d	A_data ch5 of bx+0		
1	0	0	d	d	d	d	A_data ch6 of bx+0		
1	0	0	d	d	d	d	A_data ch7 of bx+0		
1	0	0	d	d	d	d	B_data ch0 of bx+0		
1	0	0	d	d	d	d	B_data ch1 of bx+0		
1	0	0	d	d	d	d	B_data ch2 of bx+0		
1	0	0	d	d	d	d	B_data ch3 of bx+0		
1	0	0	d	d	d	d	B_data ch4 of bx+0		
1	0	0	d	d	d	d	B_data ch5 of bx+0		
1	0	0	d	d	d	d	B_data ch6 of bx+0		
1	0	0	d	d	d	d	B_data ch7 of bx+0		
E	0	0	*000b*	b	b	b	End of bx	Ring addr of B_data	E00001C
E	0	0	0	0	0	0	End of bx		E000000
E	0	0	0	0	0	0	End of bx		E000000
E	0	0	0	0	0	0	End of bx		E000000
A	0	0	e	e	e	e	HEADER A	EVNr(15:0)	A000001
B	0	0	0	0	e	e	HEADER B	EVNr(23:16)	0000000
C	0	0	bx+1	b	b	b	HEADER C	Bx_in_ev/bx of fifo	C00101D
D	0	0	n	n	n	n	HEADER D	Board identifier	D00ABCD
1	0	0	d	d	d	d	A_data ch0 of bx+1		
1	0	0	d	d	d	d	A_data ch1 of bx+1		
1	0	0	d	d	d	d	A_data ch2 of bx+1		
1	0	0	d	d	d	d	A_data ch3 of bx+1		
1	0	0	d	d	d	d	A_data ch4 of bx+1		
1	0	0	d	d	d	d	A_data ch5 of bx+1		
1	0	0	d	d	d	d	A_data ch6 of bx+1		
1	0	0	d	d	d	d	A_data ch7 of bx+1		
1	0	0	d	d	d	d	B_data ch0 of bx+1		
1	0	0	d	d	d	d	B_data ch1 of bx+1		
1	0	0	d	d	d	d	B_data ch2 of bx+1		
1	0	0	d	d	d	d	B_data ch3 of bx+1		
1	0	0	d	d	d	d	B_data ch4 of bx+1		
1	0	0	d	d	d	d	B_data ch5 of bx+1		
1	0	0	d	d	d	d	B_data ch6 of bx+1		
1	0	0	d	d	d	d	B_data ch7 of bx+1		
E	0	0	*000b*	b	b	b	End of bx+1	Ring addr of B_data	E00001E
E	0	0	0	0	0	0	End of bx+1		E000000
E	0	0	0	0	0	0	End of bx+1		E000000
E	0	0	0	0	0	0	End of bx+1		E000000
A	0	0	e	e	e	e	HEADER A	EVNr(15:0)	A000001

B	0	0	0	0	e	e	HEADER B	EVNr(23:16)	0000000
C	0	0	bx+2	b	b	b	HEADER C	Bx_in_ev/bx of fifo	C00201F
D	0	0	n	n	n	n	HEADER D	Board identifier	D00ABCD
1	0	0	d	d	d	d	A_data ch0 of bx+2		
1	0	0	d	d	d	d	A_data ch1 of bx+2		
1	0	0	d	d	d	d	A_data ch2 of bx+2		
1	0	0	d	d	d	d	A_data ch3 of bx+2		
1	0	0	d	d	d	d	A_data ch4 of bx+2		
1	0	0	d	d	d	d	A_data ch5 of bx+2		
1	0	0	d	d	d	d	A_data ch6 of bx+2		
1	0	0	d	d	d	d	A_data ch7 of bx+2		
1	0	0	d	d	d	d	B_data ch0 of bx+2		
1	0	0	d	d	d	d	B_data ch1 of bx+2		
1	0	0	d	d	d	d	B_data ch2 of bx+2		
1	0	0	d	d	d	d	B_data ch3 of bx+2		
1	0	0	d	d	d	d	B_data ch4 of bx+2		
1	0	0	d	d	d	d	B_data ch5 of bx+2		
1	0	0	d	d	d	d	B_data ch6 of bx+2		
1	0	0	d	d	d	d	B_data ch7 of bx+2		
E	0	0	*000b*	b	b	b	End of bx+2	Ring addr of B_data	E000020
E	0	0	0	0	0	0	End of bx+2		E000000
E	0	0	0	0	0	0	End of bx+2		E000000
E	0	0	0	0	0	0	End of bx+2		E000000
F	F	F	F	F	F	F	END of RECORD		FFFFFFF
I	I	I	I	I	I	I	IDLE	Between records	555AAAA

5.2 RESET LOGIC

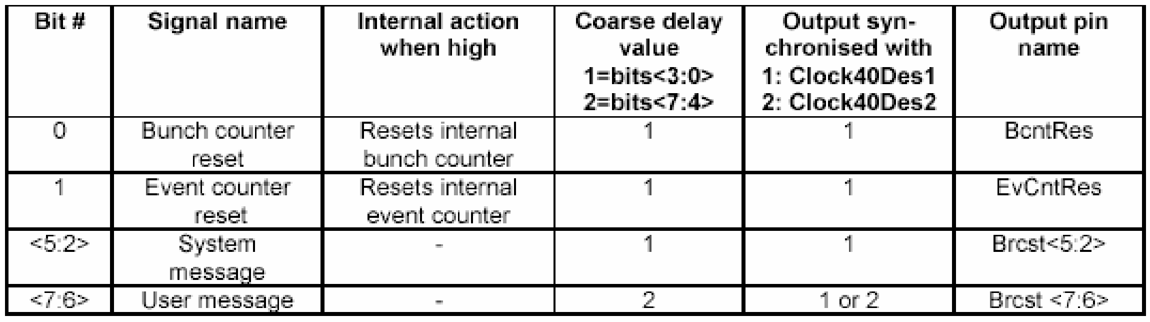
L1Res:

- **Clears FIFOs** (derandomizing buffers)
  L1Res either from backplane or from VME clears the FIFO content so that the ROC Readout Controller stays in IDLE mode after having finished the current event.

5.3 ROP logic

- ROC State Machine cannot be reset, it returns always to IDLE state. ROC starts only when FIFO is not empty and PSB is READY.

TTCrx receives BGo commands and sends them via the ROBUS to the boards.



Bit #	Signal name	Internal action when high	Coarse delay value 1=bits<3:0> 2=bits<7:4>	Output syn-chronised with 1: Clock40Des1 2: Clock40Des2	Output pin name
0	Bunch counter reset	Resets internal bunch counter	1	1	BcntRes
1	Event counter reset	Resets internal event counter	1	1	EvCntRes
<5:2>	System message	-	1	1	Brcst<5:2>
<7:6>	User message	-	2	1 or 2	Brcst <7:6>

TIM CHIP V1004, V1005 from July 2004

The TIM board sends via the RO bus to all boards

- BGo commands from TTCrx or VME on TIM
- Monitoring Request Identifier

- Test data that where loaded into the TIM register.

ROBUS	**BGO cmds**	**Monitoring**	**Tests**	
RDRQST	1	1	1	OR _STROBES
STROBE 2	0	0	1	TEST_STROBE
STROBE 1	0	1	0	MON_RQST_STROBE
STROBE 0	1	0	0	BGO_CMD_STROBE
BX 11	USER_MSG3	MON_RQST_ID 11	TEST 11	
BX 10	USER_MSG2	MON_RQST_ID 10	TEST 10	
BX 9	USER_MSG1	MON_RQST_ID 9	TEST 9	
BX 8	USER_MSG0	MON_RQST_ID 8	TEST 8	
BX 7	0	MON_RQST_ID 7	TEST 7	
BX 6	STOP_RUN	MON_RQST_ID 6	TEST 6	
BX 5	START_RUN	MON_RQST_ID 5	TEST 5	
BX 4	RES_ORBITNR	MON_RQST_ID 4	TEST 4	
BX 3	HARD_RES	MON_RQST_ID 3	TEST 3	
BX 2	PRIVATE_ORBIT	MON_RQST_ID 2	TEST 2	
BX 1	PRIVATE_GAP	MON_RQST_ID 1	TEST 1	
BX 0	TEST_ENABLE	MON_RQST_ID 0	TEST 0	

5.4 VME access timing in PSB chip_V0012

- ***The PSB chip requires that WR_PSB, VADDR and VDATA are applied at least 1 tick=25ns before EN_PSB.***
- ***EN_PSB must not be removed before NDTACK_PSB has been applied.***
- ***New EN_PSB should not be applied until NDTACK_PSB has been removed.***

Write into memory:

Begin of EN_PSB to begin of vme_we_spy (1T-puls): 3 ticks = 3 FF
Begin of EN_PSB to begin of NDTACK_PSB: 4 ticks = 4 FF
End of EN_PSB to end of NDTACK_PSB: 3 ticks = 3 FF

Read from Memory:

Begin of EN_PSB to begin of VDATA: 4 ticks = 4 FF
Begin of EN_PSB to begin of NDTACK_PSB: 6 ticks = 6 FF… *(= 75 ns after data)*
End of EN_PSB to end of VDATA: 1 ticks = 1 FF
End of EN_PSB to end of NDTACK_PSB: 3 ticks = 3 FF

Write into register:

Begin of EN_PSB to begin of write pulse (1T-puls): 3 ticks = 3 FF
Begin of EN_PSB to begin of NDTACK_PSB: 4 ticks = 4 FF
End of EN_PSB to end of NDTACK_PSB: 3 ticks = 3 FF

Read from register:

Begin of EN_PSB to begin of VDATA: 4 ticks = 4 FF
Begin of EN_PSB to begin of NDTACK_PSB: 6 ticks = 6 FF… *(= 50 ns after data)*
End of EN_PSB to end of NDTACK_PSB: 3 ticks=3 FF

Remark:

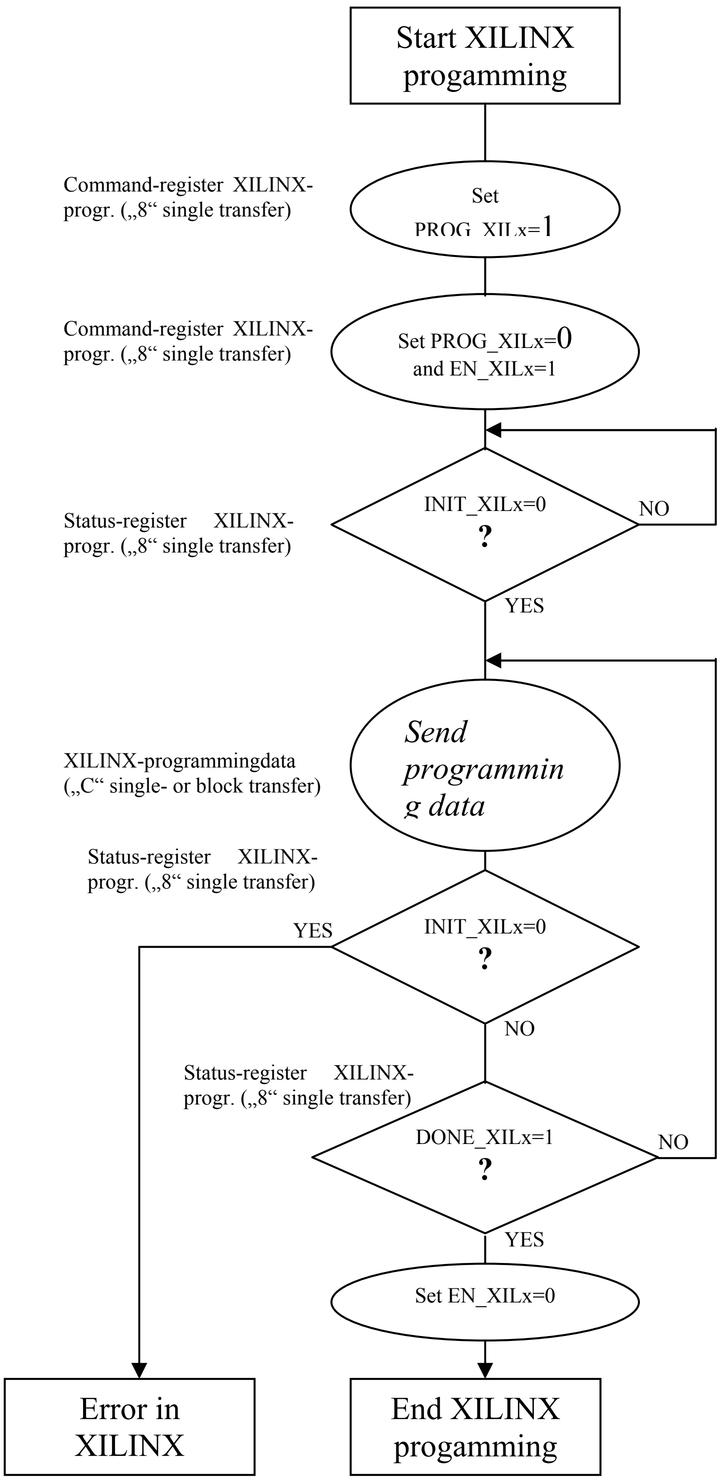
EN_PSB is delayed by an additional FF so that new VADDR and VDATA are really there when needed. This delay increases the response time for NDTACK and VDATA by 25 ns.

All VME in- and output signals of the PSB chip are registered.

Internal vme_wr is latched to keep it until end of internal (delayed) vme_en avoiding a write pulse at end of vme cycle.

6 Flowchart of XILINX-programming

The XILINX chips on PSB-card are programmed via VME-instructions, this programming sequence has to happen every time after power up. There is no PROM on board for power-up-programming! The following instructions should be implemented in the control software.


Start XILINX progamming
Command-register XILINX-progr. („8“ single transfer)
Set PROG_XILx=1
Command-register XILINX-progr. („8“ single transfer)
Set PROG_XILx=0 and EN_XILx=1
Status-register XILINX-progr. („8“ single transfer)
INIT_XILx=0 ?
NO
YES
XILINX-programmingdata („C“ single- or block transfer)
Send programmin g data
Status-register XILINX-progr. („8“ single transfer)
INIT_XILx=0 ?
YES
NO
Status-register XILINX-progr. („8“ single transfer)
DONE_XILx=1 ?
NO
YES
Set EN_XILx=0
Error in XILINX
End XILINX progamming

Beginning of configuration in enabled XILINX chip(s).

Checking whether XILINX chip(s) is (are) ready for configuration.

Sending programming data of XILINX chip(s) on D0. Programming data file comes from XILINX software. Setup-file has to implement these data and send it via single- or blocktransfer.

Checking errors during programming of XILINX chip(s).

Checking end of programming data frames of XILINX chip(s).

Ending programming sequence of XILINX chip(s).

After the XILINX programming sequence all other VME-accesses on card are possible, e. g. DPM-access or FIFO-access in XILINX chips and so on.