Pre-Processor Asic
User and Reference Manual

Version 1.1

D. Husmann, M. Keller, K. Mahboubi, C. Schumacher

November 29, 2002

Contents

List of Figures

List of Tables

Introduction

The Pre-Processor Asic (PPrAsic) is the heart of the pre-processor system of the ATLAS Level-1 Calorimeter Trigger. It provides most of the fast signal processing capabilities required for preparing the analog data received from the calorimeters for the level-1 trigger processors, which select interesting physics events for later processing stages.

Figure 0.1 shows a block diagram of the PPrAsic. There are two main building blocks, the Bunch-Crossing Identification (BCID) and the Readout. Other important blocks include the serial interface used for control and readout, the synchronisation stages at the input of the chip, the playback, histogramming and rate-metering, bunch-crossing multiplexing and output summing stages.

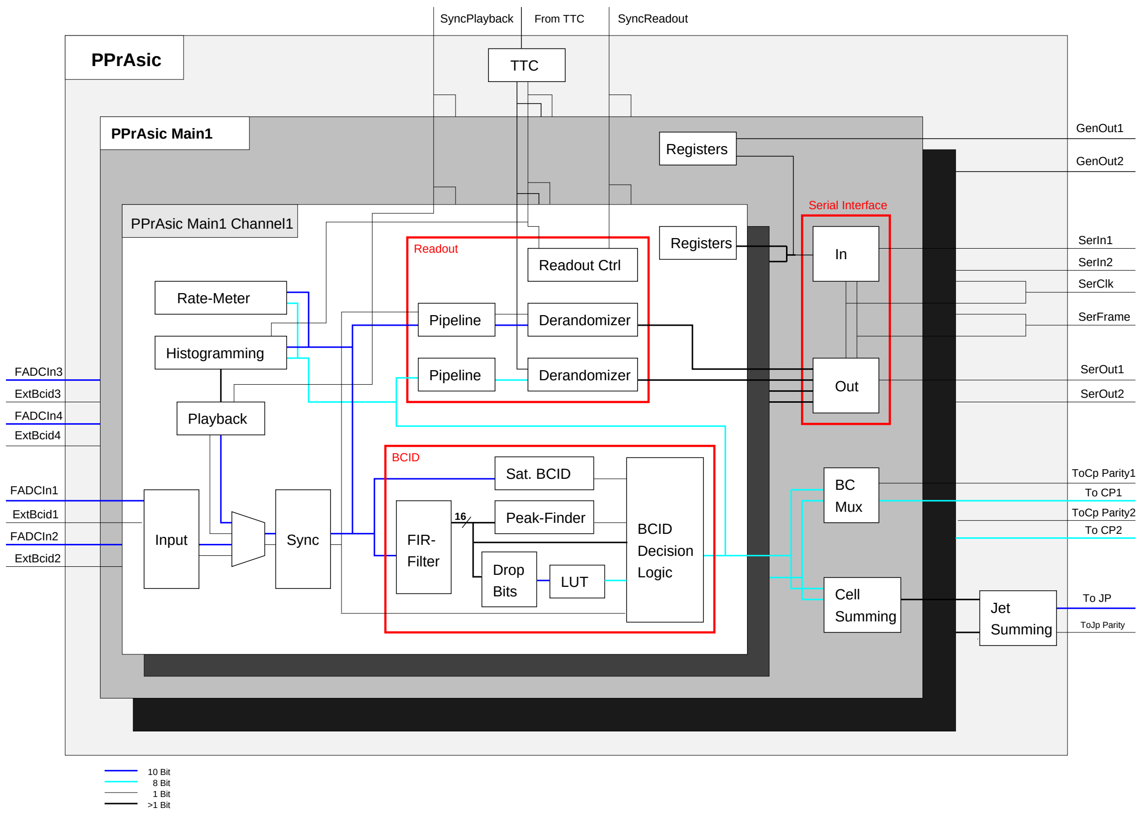


Figure 0.1: Block diagram of the Pre-Processor Asic

Chapter 1

Bunch-Crossing Identification

The Bunch-Crossing Identification (BCID) is done by two different algorithms on the PPrAsic. One is optimized for the identification of saturated pulses, the other one for non-saturated pulses. As shown in the block diagram both paths are built in parallel and are always active. There is also the possibility to use an external bunch-crossing identification. The decision which of the algorithms is used for the final result, is made in a block called *BCID decision logic*. This block will be explained in section 1.3.1.

1.1 Bunch-Crossing Identification for saturated pulses

Inputs that can be determined from outside:
→ RegSatHigh
→ RegSatLow
→ RegSatLevel

There are three programmable inputs to the BCID block for saturated pulses. First the level for saturation (*RegSatLevel*), second and third the two levels for the algorithm itself (*RegSatHigh* and *RegSatLow*). The first input is used to decide whether the pulse is saturated or not and to activate the algorithm. When the data value coming from the ADC reaches or goes beyond the level that was chosen for saturation, the algorithm is performed once. It is rearmed by the next following sample that is under the saturation level. To have a programmable saturation level it may be useful to increase the overlap region between the two algorithms (saturated and non-saturated). Once the algorithm is activated, it looks on the leading edge of the pulse shape, because the origin of the pulse is the only fixed attribute that defines the interaction time point. To get information about the leading edge of the pulse two programmable thresholds are used. These threshholds are used to compare them with the pulse values two bunch-crossings before saturation. This way it is possible to get information about the steepness of the leading edge of the pulse. For the reason to save some unnecessary flip-flops, the incoming data are compared directly with the two thresholds (RegSatHigh and RegSatLow) and just the 1-bit result is stored.

1.2 Bunch-Crossing Identification for non-saturated pulses

The identification of the right bunch-crossing for non-saturated pulses is done by the combination of two logic blocks. First the incoming data is leaded to a FIR-filter that is used to sharpen the pulse. Then in the second step a peak-finder is used to get the excact bunch-crossing where the peak of the pulse is located. As shown in the block-diagram of the PPrAsic the FIR-filter is not

$t_{sat} \geq RegSatLevel$	$t_{sat-2} > RegSatLow$	$t_{sat-1} > RegSatHigh$	t_{sat}	t_{sat+1}
0	X	X	0	0
1	1	1	1	0
1	0	1	0	1
1	X	0	0	1

Table 1.1: Logic table to be implemented in the Preprocessor ASIC. Fulfilled threshold conditions are indicated by a logic '1', the identified bunch-crossing has a logic '1' at t_{sat} or t_{sat+1}.

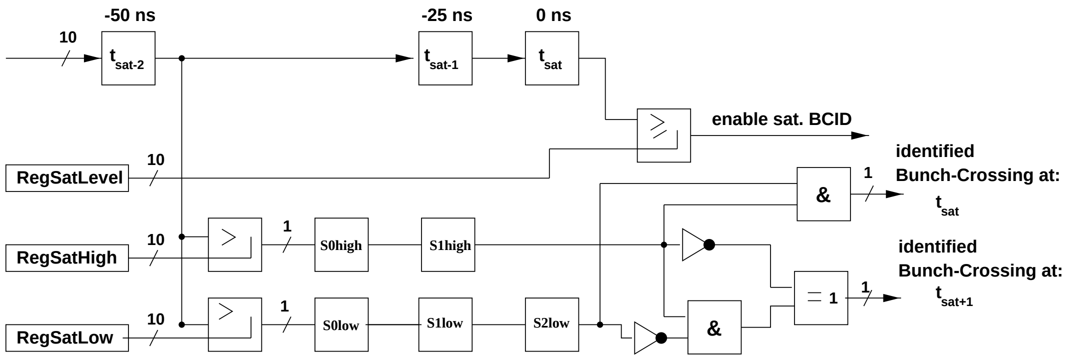


Figure 1.1: Schematic implementation of the saturated BCID logic. The identified time slice is indicated to the BCID decision logic. Timing synchronization of the result bits and the re-arming of the algorithms are not shown.

only used as a preparation of the pulse for the following peak-finder. Its other functionality gets clearer in the desciption of the BCID-decision logic.

1.2.1 The FIR-filter

Inputs that can be determined from outside:
→ RegFIRCoeff1
→ RegFIRCoeff2
→ RegFIRCoeff3
→ RegFIRCoeff4
→ RegFIRCoeff5

The FIR-filter has five coefficients that are all 4-bit wide. Due to its function as a sharpening filter the values of these have different meanings. The first and the fifth coefficient can be used as a positive or a negative number with a range of -7 up to +7. The range is not of -8 up to +7 as it is expected for a 4-bit signed multiplication, because the multiplication is divided into two clock cycles. The reason and the implementation is explained later. The other three values are used only as positive numbers (range 0 to 15). Choosing the right coefficients is essential for the correct BCID done by the peak-finder. For that reason they have to be programmed individually for each channel to comply with different pulse-shapes. The first idea of the implementation was to store the incoming data into five registers, multiply them with the coefficients and then sum them up.

To be able to use the short time that is forseen for the FIR-filter more efficient the whole stategy was changed. In this new strategy incoming data are directly multiplied with the five different coefficients, added with results from the previous multiplication and then stored into four 17-bit wide registers. This strategy needs more area, but has a better performance what timing concerns.

Because of the complexity of the multiplication it seemed to be impossible to do the multiplication and the final summing in one clock cycle. So it was decided to split the multiplication in two parts and to reserve one clock cycle for the first part and to use another clock cycle for the second part and the final summing. In case of the first and the fifth coefficient the first part is a 10 x 3 bit multiplication and the second part is the converting into the two‘s complement. This way the first bit of the coefficients can only be used to sign if the coefficient is positiv or negative and therefore the the range is -7 up to +7. For the other coefficient the first part is used for two 2 x 10 bit multiplications in parallel and the second part is used to sum up the results.

For more details about the implementation it is advisible to look at the detailed structure of the FIR-filter shown in figure1.2.

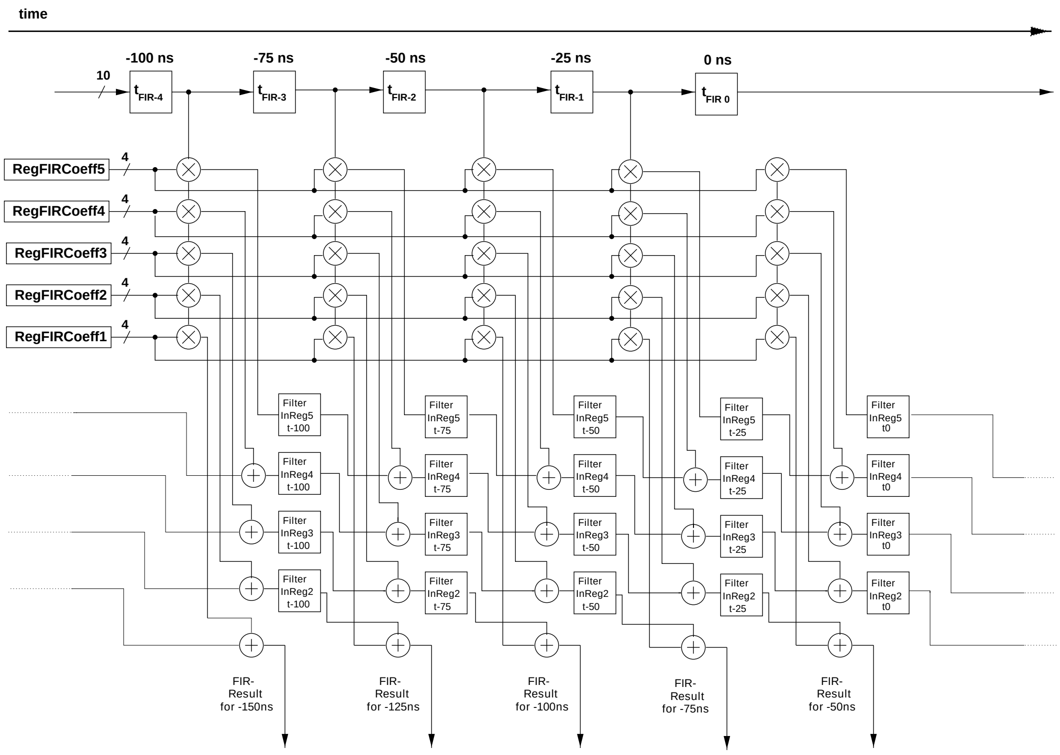


Figure 1.2: Schematic implementation of the FIR-filter

1.2.2 The peak-finder

Inputs that can be determined from outside:

$\rightarrow$ RegPeakFinderCond

The strategy of the peak-finder is to compare for each bunch crossing the incoming data with the data from the previous and the following bunch-crossing. For this comparison there are two possible conditions to get a positive result. These two conditions are determined through the RegPeakFinderCond bit as the following:

RegPeakFinderCond $= 0 \Rightarrow$ the condition is $bc(t-1) < bc(t0) \geq bc(t1)$.

RegPeakFinderCond $= 1 \Rightarrow$ the condition is $bc(t-1) < bc(t0) > bc(t1)$.

1.3 Other modules in the BCID-block

1.3.1 Bunch-Crossing Identification Decision logic

Inputs that can be determined from outside:
→ RegBcidDecision1,RegBcidDecision2,RegBcidDecision3
→ RegDelayExtBcid
→ RegEnergylevelLow
→ RegEnergylevelHigh

The task of the BCID Decision Logic is to allocate the various BCID algorithms to energy ranges. The energy range is 16 bit wide and can be divided into three regions using the two thresholds RegEnergylevelLow and RegEnergylevelHigh. The registers are just 10 bit wide but are expanded up to 16 bit by shifting them and by filling zeros into the lowest 6 bits. The value of 16 bit is necessary, because for the setting of the energy range a comparison between the thresholds and the output data of the FIR-filter is done. For each of the three energy regions a function can be set by using the programmable inputs RegBcidDecision1 up to RegBcidDecision3. These inputs are 8 bit wide numbers that contain all the information about which input combination built out of the three BCID algorithms leads to which result. In table 1.2 the building of these numbers is shown as well as an example for a setting that uses only the algorithm for saturated pulses.

The energy regions are based either on the FIR filter output or the direct ADC input depending on the setting of register *DecisionSource*.

The BCID result is overridden by the value of the register *SaturationValue*, if the *SatOverride* register of the energy region triggering the BCID decision is set.

For example: if the peak-finder result is 1, the result of the saturated algorithm is 0 and the external BCID says 1, the fifth bit of the RegBcidDecision for each energy range has to be set. If it is set to 1 it means that for this combination the decision logic output will be 1.

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Bit No.**
x	x	x	x	-	-	-	-	Peak-Finder
x	x	-	-	x	x	-	-	Sat-Bcid
x	-	x	-	x	-	x	-	Ext.-Bcid
1	1	1	1	0	0	0	0	F0 only Peak-Finder
1	1	0	0	1	1	0	0	CC only Sat-Bcid
1	0	1	0	1	0	1	0	AA only Ext.-Bcid

Table 1.2: Table that shows for each combination of the results of the algorithms the correct setting and the number of the according bit in the RegBcidDecision-number.

1.3.2 The Drop-Bits module

Inputs that can be determined from outside:
→ RegStartBit

This module is used to reduce the 16 bit wide output of the FIR-filter to the 10-bit that are used as an input of the Look-Up-Table. Dropping the lower bits causes a reduction in resolution. Dropping the MSBs can lead to serious corruption of the energy values. The idea of the programmable start-bit is based on the fact that for almost all the settings of the FIR-filter coefficients the 16-bit can not be reached. What has to be done is to calculate for each setting of the FIR-filter the highest result that is possible and then choose the correct start-bit belonging to this configuration. For example: if the FIR-filter has the coefficients -1, 0, 5, 0, -1 the highest possible result is $1023 * 5 = 5115$. So the maximum result is smaller than 13 bit (8192) and therefore it is possible to drop the three highest bits without loosing any information $\Rightarrow$ RegStartBit can be set to 3.

The drop bits module makes sure that the output value is not corrupted by dropping more significant bits than the selected ones. If one of these bits is set a saturated value is put out, i.e. the maximum possible value.

1.3.3 The Look-Up-Table

Inputs that can be determined from outside:

$\rightarrow$ RegBypassLut

The Look-Up-Table is used for fine calibration of the digitised data after the peak-sharpening FIR-filter to the deposited transverse energy E_T. The incoming 10-bit data after the drop-bits module are mapped to 8-bit used as the BCID output. This can be done using a 1024*8-bit memory or by just dropping the two least significant bits. To be able to chose between this two options the programmable register RegBypassLut is implemented.

RegBypassLut $= 0 \Rightarrow$ the 1k*8-bit memory is used.

RegBypassLut $= 1 \Rightarrow$ the two least significant bits are dropped.

The Look-up-Table can be used to substract a pedestal and it can apply a minimum threshold in order to suppress noise.

Chapter 2

Readout

The PPrAsic provides facilities to read out the raw trigger data as being processed by the Pre-Processor Asic, on which the trigger decision of succeeding processors is based. This is required to monitor the function of the trigger and to verify that the trigger data and the data from the seperated path of full-granularity readout of the calorimeters are consistent.

2.1 Event Data

Two types of data can be read out, raw digitized data and BCID results. The first type of data is directly read from the PPrAsic input, the second type is read after BCID and energy calibration and is identical to the data transmitted to the level-1 trigger processors before bunch-crossing multiplexing and jet cell summation.

Data to be read out is stored to two separate pipeline memories at full clock rate. The length of the pipelines is 128 words each, corresponding to a time of 3.2 μs. When an event was accepted by the Central Trigger Processor (CTP) of the Level-1 Trigger it sends a signal, the *L1Accept*, to all systems that provide readout data using the TTC system [1]. On reception of this signal the PPrAsic copies the data belonging to this event from the pipeline memory to a buffer, the *derandomizer*, which holds the data until it is read out via the serial interface of the PPrAsic. The length of the derandomizer is 128 words. It is used to compensate statistical fluctuations of the Level1Accept signal, when its frequency is temporarily higher than the maximum frequency to read out events from the PPrAsic. The CTP makes sure that the average frequency of the L1Accept signal never exceeds 100 kHz.

For each event a configurable number of samples can be read out. These numbers are set in the registers *NumBcRaw* and *NumBcBcid*. In standard readout mode five samples of raw data are read out to provide some information of the analog pulseform and one sample of BCID result data, which is the input data to the trigger processors and can be used to check the correct function of these components.

2.2 Header Word

In addition to the event data a header word is read out for each event. This header word contains the lowest four bits of event and bunch-crossing number as provided by the TTC system. These numbers are used to check the synchronicity of all readout components. Figure 2.1 shows the format of the event header word.

The event and bunch-crossing numbers are periodically reseted to zero by the external signals *EvCntRes* and *BcCntRes*. These synchronous reset signal are provided by the ATLAS Trigger,

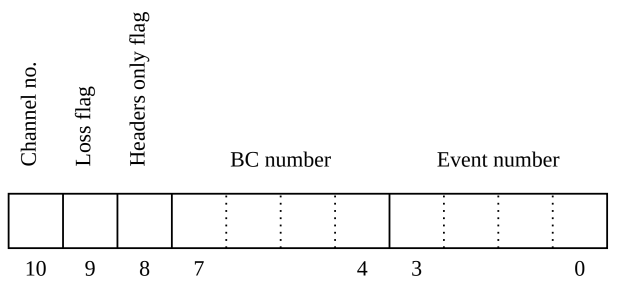


Figure 2.1: Format of the event header word added to the PPrAsic readout data

Timing and Control system (TTC) [1]. After an asynchronous reset by the *ResetBar* signal has occured, the counters remain inactive at value 0 until the next synchronous counter reset signal takes place.

The header word also contains information about the PPrAsic channel the event belongs to, and the state of the readout. There are two bits indicating special states of the readout, the *HeadersOnly* bit is set when only headers are written to the derandomizers, event data are deleted. This is the case, when one of the derandomizers is almost full, but enough space exists to write header words. If one of the derandomizers is completely full, events are lost completely and the *LossFlag* is set. Both operating states should only occur in case of problems with readout and indicate a serious malfunction of the system.

Two channels of the PPrAsic are read out by one serial interface. The read out data, consisting of 11 bit words is mapped on the 13 bit wide serial word structure by adding flag bits and interleaving with readback words containing status or configuration data. Details about the serial word format and readout data mapping can be found in section 6.

Chapter 3

Histogramming and Rate-Metering

The PPrASIC is capable of producing detailed information on the raw, i.e. digitized, and/or processed, i.e. BCID calibrated, signals. The monitoring is – upon request, which is done by setting the appropriate registers – performed by producing histograms and rate measures for each channel. The histograms are written into memories, whereas the rate information is stored in buffers. The content of these memory blocks and buffers is then read back during the readout of the chip. Each channel (or trigger tower) signal could be quasi-online monitored by the receiving processors for diagnostics. In the following a description is given on the implementation of the histogramming and rate-metering blocks. For more information on this chapter see reference [2].

3.1 Histogramming

The Histogramming block is implemented as a finite state machine with a passive and an active mode. When enabled[1] the histogramming will enter the active mode, whenever the contents of the playback memory is read back and/or is cleared. In active mode, based on the setting of the '**RateSource**' register, either the FADC or the BCID signal is selected for further processing. No buffering is performed on these inputs. At this stage is also the actual bunch-crossing number checked against the allowed bunch-crossing window defined by the lower and upper limit registers

'**HisLowerBcH**', '**HisLowerBcL**' $\Longrightarrow$ Lower Bunch-crossing number.

'**HisUpperBcH**', '**HisUpperBcL**' $\Longrightarrow$ Upper Bunch-crossing number.

in order to decide whether the input should be histogrammed. The histogram itself is realized with a single-ported memory block with an 8-bit address and an 11-bit data bus. The same memory can alternatively be used as memory for playback data. This gives a 256 channel long histogram, with 11 bits wide channels. The range and resolution of this histogram is set by the two '**HisOpMode**' and '**RateSource**' register combination. Each memory location would serve as a channel of the histogram containing the number of entries into that location. All signals above a threshold set by the '**HisEtThresh**' register are histogrammed. The complete cycle would be as follows:

- Based on the input, the channel number which should be incremented, is determined. This channel number is put on the memory address lines.

[1]As explained in chapter 9 histogramming is only enabled when '**PlaybackEnable**' and '**HisEnable**' register combination is set to '0' and '1' respectively.

- A memory read operation is performed. This takes four clock ticks. No signal is processed during this period. At the end of the period the content of the corresponding channel is incremented and at the same time checked for all 1's, since this would indicate that a next entry in this channel would push the content into overflow. An 'Overflow' flag is set if this would be the case.
- The new channel content and the channel number (which obviously is the same as for the corresponding read operation) are put on the memory data and address lines, respectively. A memory write operation is then initiated. This write operation won't be stopped even if the 'Overflow' flag is set. This is simply because the content is still not in overflow.

This cycle continues until one of the channels goes into 'Overflow'. This would in turn mean that the channel histogram is filled. Whenever this is the case, the histogramming module signals it to the rest of the chip by asserting the 'HisReady' signal high. At the same time the state machine enters the passive mode. Further operation is stopped until the content of the channel memory is read back and subsequently cleared. Upon receiving of the 'memory cleared' signal the histogramming block enters the active mode and the cycle starts from the beginning.

3.2 Rate-Metering

The Rate module is functionally divided into two submodules:

'**RateMeter**' performs the measurement of relevant quantities required for offline rate calculation by the receiving processors.

'**RateReadback**' is responsible for the readout of the quantities provided by the **RateMeter** submodule.

In the '**RateMeter**' submodule, based on the '**RateSource**' register setting, it is determined whether the raw FADC or the calibrated BCID signals should be considered for rate measurements. The setting of the '**RateEnable**' register indicates whether or not the rate measurements should be active. When active, signals are checked against the content of the energy threshold register, called '**RateEtThresh**'. A 20 bit counter keeps track of the number of received signals in excess of the chosen threshold. The number of bunch crossings, within which the channel signal is compared against the energy threshold register, is counted in blocks of 1024 bunch crossings (i.e. 25.6 μs time blocks). This is achieved by a 10 bit and a 16 bit counter combination, in that each time the 10 bit counter wraps around, the 16 bit counter is incremented. The contents of the 20 bit signal counter, containing the number of signals above a given threshold, and the 16 bit time block counter, containing the number of 1023 bunch crossing long time blocks, is output to the '**RateReadback**' submodule. The counting stops either when the signal counter overflows or when the content of the time block counter exceeds the content of the '**RateDelTime**' register – called overflow and timeout condition respectively. At this stage the readyness of the rate related quantities is signaled to the '**RateReadback**' submodule by setting a dedicated signal line high. When this logic 1 is sampled, signal and time quantities are stored into dedicated registers.

The ratemeter readback generates a readback block of four words, containing the time and count information. This information consists of a 20-bit `TimeInterval` and a 16-bit `Count`. Figure 3.1 shows the exact format of this words. It has to be interpreted as: In time `TimeInterval`$\dot{2}5\mu s$ `Count` bunch crossings carried an energy value greater than the value of register `RateEtThresh`.

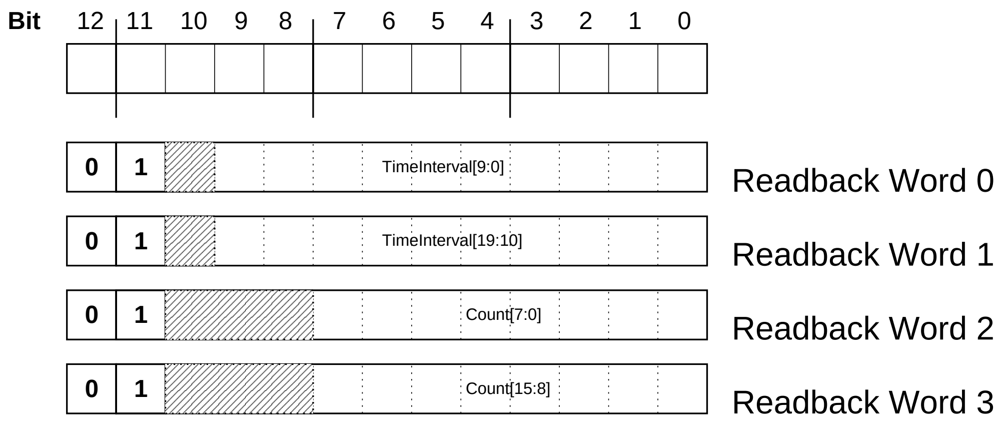


Figure 3.1: Format of ratemeter readback data block

Chapter 4

Control logic

The PPrAsic is controlled via its serial interface, which is described in chapter 6. One chip contains two serial interfaces, each operating on two trigger channels. These interfaces can be operated independently or can be daisy-chained to minimize the number of I/O signals. Control commands and configuration data are sent to the chip, status information, register and memory contents are read back from the chip. The protocol of the serial interface is based on 13 bit wide words. How these words are used for configuration and control is described in section 4.1. The words which are used for readback data are contained in the general output format, which interleaves event readout data and readback data using a fixed scheme. That scheme is described in chapter 6 about the serial interface. The format of readback words and how to operate readback is described in section 4.2.

4.1 Configuration

Two types of words can be sent to the PPrAsic, command words which toggle actions and data words, which contain data to be written to internal blocks of the chip, like memories or registers. The two types are identified by the value of the most significant bit of the word. The figures 4.1 and 4.2 show the format of the input words for the two different types. Loading of registers and memories is described in section 4.1.1. This involves the command *CfgAddress* and the different data words. The function of the other commands is explained in section 4.1.2.

4.1.1 Loading of memories and registers

Each trigger channel of the PPrAsic has the same set of registers and memories, all of them get an individual address. When sending data to one of these components the address of the component has to be set and then the succeeding data sent to the chip is loaded into the component.

The address is specified by the *CfgAddress* command. It takes three fields as arguments, a *global address*, a *local address* and a *memory bit*. These three address fields are stored inside the chip and remain valid until overwritten by a new address as result of sending another *CfgAddress* commands.

Memory Bit

If the memory bit is set to 1, the two addresses are interpreted as address of a memory block, if the memory bit is set to 0, the two addresses are interpreted as register address.

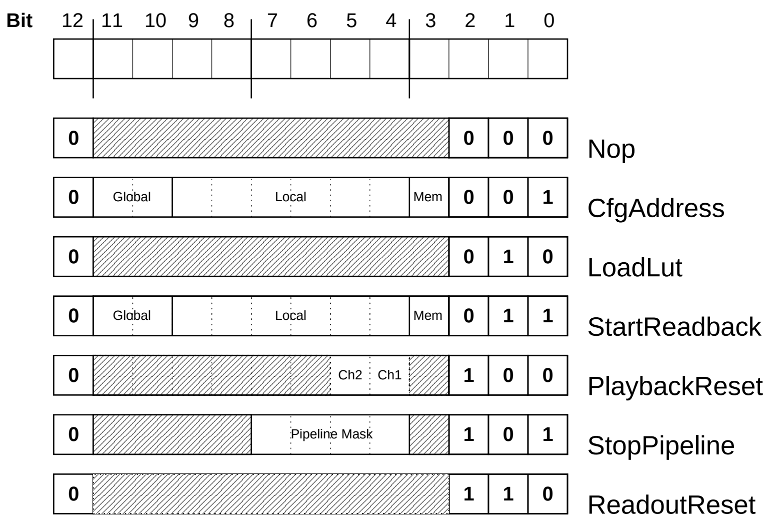


Figure 4.1: Format of PPrAsic commands

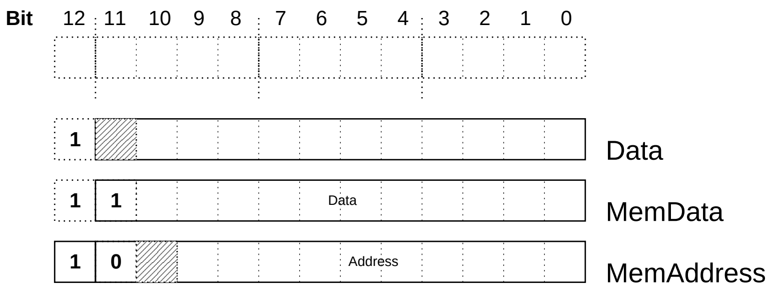


Figure 4.2: Format of input data words of PPrAsic configuration protocol

Local Address	Name of Component
1	Look-Up-Table
2	Playback/Histogramming Memory
3	Pipeline Memory for raw data
4	Pipeline Memory for BCID results and event headers
5	Ratemeter (read only)

Table 4.1: Local addresses of memory blocks

Global address

The global address, consisting of two bits, indicates which of the two trigger channels connected to a serial interface is targeted by the local address. If the lower bit is set to 1 (global address 1), channel 0 is addressed, if the higher bit is set to 1 (global address 2), channel 1 is addressed. This means that if both bits are set to 1 (global address 3), both channels are addressed, and the same data is written to both channels simultanously.

Local address

The local address gives the number of the register as listed in section 9 or the number of the memory block as listed in table 4.1. The ratemeter is a special case, which can not be a target for a configuration operation, but its local address is used for readback. See section 3.2 for details.

Memory loading

When a memory block is selected as target of a configuration operation, bit 11 of configuration data words is interpreted as flag, differentiating two types of memory data words, *MemData* words (flag 1) and *MemAddress* words (flag 0). See figure 4.2 for the exact format of these words.

For configuration of the memories there exists an address counter for each memory. It is set to address 0 after reset. When a *MemData* word is received by a memory block, it stores the data contained in this word to the memory at the location the address counter points to. Then the address counter is incremented by 1. This means that a sequence of *MemData* words is stored to the memory as a continous block of data and that by sending a number of *MemData* words equal to the number of memory locations the memory is completely loaded.

For defining the start address where a data block is to be stored or to write individual memory locations, the *MemAddress* word is used. It sets the address counter of the targeted memory to the value given as argument to the *MemAddress* word. Data received with *MemData* words is then stored starting with this new address.

4.1.2 Command Reference

Commands are indicated by bit 12 of the serial word set to 0. The exact format of the commands is shown in figure 4.1. The function of the comands is explained in this section.

Nop

This command intentionally does nothing.

CfgAddress

Set destination of suceeding data words. See section 4.1.1 for details.

Bit	Pipeline
4	Raw data, channel 1
5	BCID result data, channel 1
6	Raw data, channel 2
7	BCID result data, channel 2

Table 4.2: Correspondence of *StopPipeline* command argument and pipeline memories

LoadLut

Start internal loading of Look-Up-Table. The LUT is loaded with the number of zeroes given by register *LutPedestal* starting at memory address 0. The remaining memory locations are filled with a ramp starting at zero and incrementing for each memory location by a value corresponding to register *LutSlope*. The value stored to the register has to be divided by 256 to get the effective value of the increment. Only the integer part of the ramp value is written to the LUT. The LUTs of both channels connected to the serial interface are loaded, when this command is received.

StartReadback

Start a readback operation. See section 4.2 for details.

PlaybackReset

Reset read address counter of playback memory. The argument identifies, which channels are reset. If bit 4 of the command word is set, channel 1 is reset, if bit 5 is set, channel 2 is reset.

StopPipeline

Stop readout pipelines. After execution of this command the pipelines denoted by the argument are stopped. This means that no more data is written to the pipeline memory and readout data may be lost. Pre-Processing data for the level-1 trigger processors is not affected. The argument of this command contains a mask of the pipelines to be stopped (see table 4.2). To allow synchronous stopping of pipelines, the first serial interface of the PPrAsic controls the stopping of all four trigger channels. The second serial interface does not process this command.

ReadoutReset

Synchronously reset readout logic of both channels connected to a serial interface. The pipeline memories are reenabled, the derandomizers are cleared and the readout logic is reset to wait for the next L1Accept. Readback of control data is not affected by this command.

4.2 Readback

All configuration data stored to registers and all memory contents can be read back through the serial interface. These readback data is interleaved with the readout data in a fixed scheme (see figure 6.3 in section 6.2 about the data format of the serial interface). One word of readback data is transmitted per readout cycle, which is used for readout of all data belonging to one event. After removing the event readout data from the stream only the readback data stream remains.

There are three types of readback words: *status* words, which are sent when no readback is requested, *header* words, which start a block of readback data and *data* words containing the actual data read back. Figure 4.3 shows the format of these words.

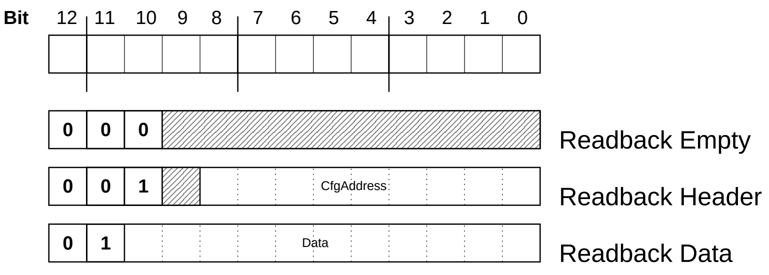
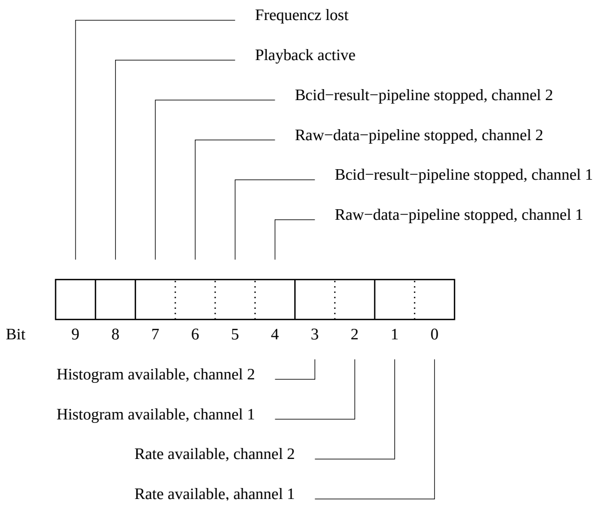


Figure 4.3: Format of readback words



Frequencz lost
Playback active
Bcid−result−pipeline stopped, channel 2
Raw−data−pipeline stopped, channel 2
Bcid−result−pipeline stopped, channel 1
Raw−data−pipeline stopped, channel 1
Bit
9
8
7
6
5
4
3
2
1
0
Histogram available, channel 2
Histogram available, channel 1
Rate available, channel 2
Rate available, ahannel 1

Figure 4.4: Format of status word

The format of the status word is shown in figure 4.4. The *Frequency Lost* bit is set, when the frequency at the *FreqIn* input of the PPrAsic is lower than 0.3 MHz (see also 7). *Playback Active* is set to 1, when one of the two channels connected to the serial interface used for readback of the status word is in playback mode. This means that the data put into the processing pipeline is taken from the internal playback memory rather from the external ADC inputs. The *Histogram Available* and *Rate Measurement Available* bits are set, when the corresponding histogram or rate measurement is ready for readback.

A readback operation is started by sending a *StartReadback* command to the PPrAsic. This command contains the address of the block to be read back as argument. The same addressing scheme as described in section 4.1.1 is used. The only difference is that the global address 3 is not a legal value for readback.

As response to this command a readback header is generated. This contains the address of the component to be read back and should be the same as the argument of the *StartReadback* command. After the header a number of data words is generated. The number depends on the component to be read back. For registers it is one, for memories the size of the memory in words. Another header or an empty word terminates the readback data block.

Chapter 5

Input and Output

5.1 ADC Input

The ADC input is adapted to the ADC AD9042 by *Analog Devices*. The highest bit is inverted by default. This can be switched off by setting the register *InvMsbDisable*.

5.2 Input Synchronisation

ADC input data can be latched with the positive or negative edge of the system clock, provided at the *Clk* input of the PPrAsic. This makes it possible to compensate for different phases of different ADC clocks and to ensure that the setup times of the ADC inputs are not violated. Registers *InBcidNegedge* and *InDataNegedge* are used to set the clock edge.

In order to compensate for different coarse delays of input data, all ADC input data can be delayed by a configurable number of system clock cycles. Registers *SyncDelayBcid* and *SyncDelayData* are used for configuration of this value.

5.3 External BCID processing

The external BCID bit can be an asynchronous signal. It is synchronized at the input of the PPrAsic with the edge of the internal clock set with register *InBcidNegedge*. It is made sure that that the signal processed internally is exactly one clock cycle active, even if the external signal is longer.

5.4 Output to Cluster Processor

Data is truncated to 8 bit per channel by the LUT. Two channels are multiplexed to one output by using the bunch-crossing multiplexing scheme described in section 5.4.1. This results to a word of a width of 8 bits and one additional flag bit. An odd parity bit is added. The value put out at the cluster processor output for all channels containing a zero-signal is `b10_0000_0000`, with the most significant bit being the parity bit.

5.4.1 Bunch-Crossing Multiplexing

Data from two neighbouring channels on a chip are multiplexed into a single output, received by the Cluster Processor. A multiplexing scheme is adopted in which the content of the two channels in the same buch crossing is sent in two consecutive clock cycles. In order to make it possible for the receiving end to uniquely identify the bunch crossing and the channel contents a flag bit is added to each channel data. An odd parity bit is also added to this data-plus-flag-bit combination for error detection purposes. So at each bunch crossing a 10-bit long signal is sent, on the output lines, to the Cluster Processor. The 8 least significant bits are the original data containing the channel content at the given bunch crossing. Bit number 9 is the flag bit and the most significant bit is the parity bit. The flag bit is set according to the following scheme:

- In the first clock cycle the flag bit indicates which channel is being sent in the current bunch crossing: $0 \to$ ch 1, $1 \to$ ch 2. Channel 1 takes precedence if both channels happen to contain non-zero signals.
- In the second clock cycle (following the previous one) the flag bit indicates to which bunch crossing the sent data actually belongs to: $0 \to$ previous, $1 \to$ current. A zero data in this second half is always accompanied with a flag bit set to 1.

This bunch crossing multiplexing cycle, containing two clock ticks, starts whenever either of the two channels contains a non-zero data. This scheme implies that the receiving end should first check the 8 LSBs for a non-zero data to start the de-multiplexing.

5.5 Output to Jet Processor

The BCID results of all four channels of a PPrAsic are summed together and put out to the jet processor. The 10 bit sum is truncated to a 9 bit value. This can be done in two different modes: The default setting is truncating the highest bit of the sum. If the bit, which is truncated, is set, the output sum is set to its maximum value of 511. Alternatively the lowest bit of the sum can be dropped. For both modes saturation information is preserved by setting the output sum to 511, if one of the four input channels is saturated (has a value of 255). By this measure the succeeding trigger processors retain the information about saturaton of input channels.

In a third summing mode, no truncation occurs, and the parity output is used to transmit the lowest bit of the sum and the other 9 bit are transmitted by the normal jet processor outputs.

Chapter 6

Serial interface

The Pre-Processor Asic contains a serial interface which is used for configuration and readout purposes. It has to provide enough bandwidth to read out all selected data from the processing pipeline while the trigger operates with the full rate of 100 kHz. Additional requirements concern configuration of the Asic, readback of configuration and memory data and daisy chain capability.

6.1 Architecture

The serial interface uses four data lines and two additional daisy chain signals. A synchronous protocol is used to transmit data words having a width of 13 bits. Figure 6.1 shows a block diagram of the interface.

The clock signal *SerialClk* operates the input and output shift registers. *DataIn* and *DataOut* are the input and output of the according shift register. The signal *Frame* is used to identify word boundaries by controlling the loading of input register and output shift register. Figure 6.2 shows the timing of the relevant signals.

Several serial interfaces can be connected together building a daisy chain by feeding the *Daisy-Out* signal to *DataIn* and *DataOut* to *DaisyIn* of the next chip. The frequency of the *Frame* signal then has to be divided by the number of chips in the daisy chain.

The serial clock can be operated with a different frequency than the core of the Asic. A synchronizer has to generate a signal from the *Frame* bit that indicates a stable value in the input register. Processing has to be fast enough to provide a stable output value before the next *Frame* signal occurs.

6.2 Data Format

Data from three different origins have to be read out from the PPrAsic via the serial interface. These are the two processing channels of the PPrAsic providing raw trigger data and a channel providing readback capability for parameter and status registers and memories. Data from these three channels are multiplexed to the serial output using a fixed time scheme. Two control bits indicate the type of data. Figure 6.3 shows the scheme.

One control bit is used to identify if the word is a readback word (bit value: 0) or an event readout word (bit value: 1). The second control bit is used within the readout word. It indicates if the readout word carries BCID result or event header data (bit value: 0) or raw data (bit value: 1). The first word of the BCID result/event header block is always one header word. Zero to four BCID result data words can follow, depending on the setting of register *NumBcBcid*. The raw data block can contain zero to five words depending on the setting of register *NumBcRaw*.

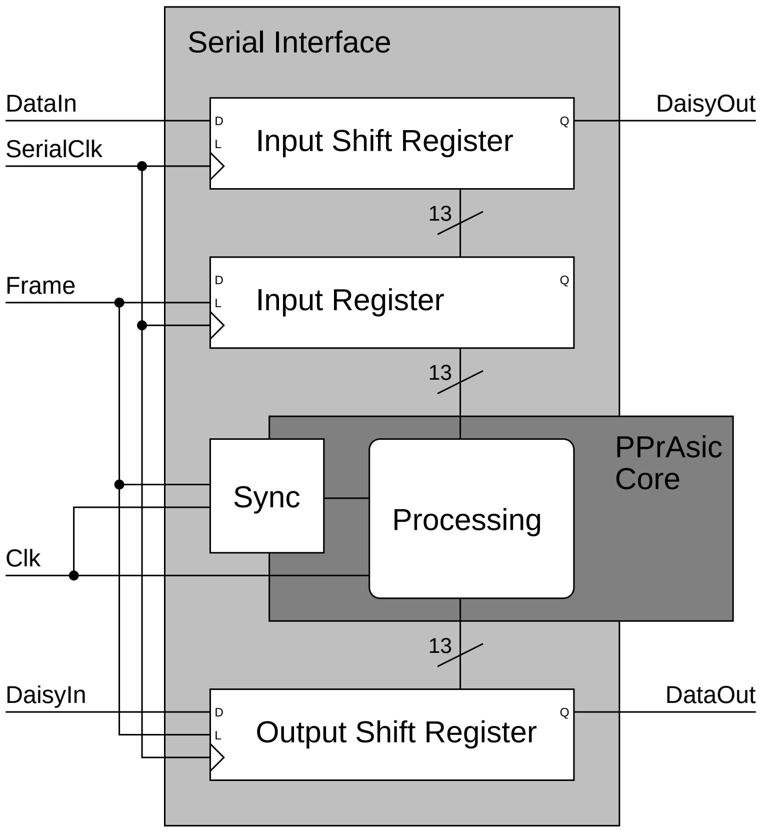


Figure 6.1: Block diagram of the serial interface of the PPrAsic

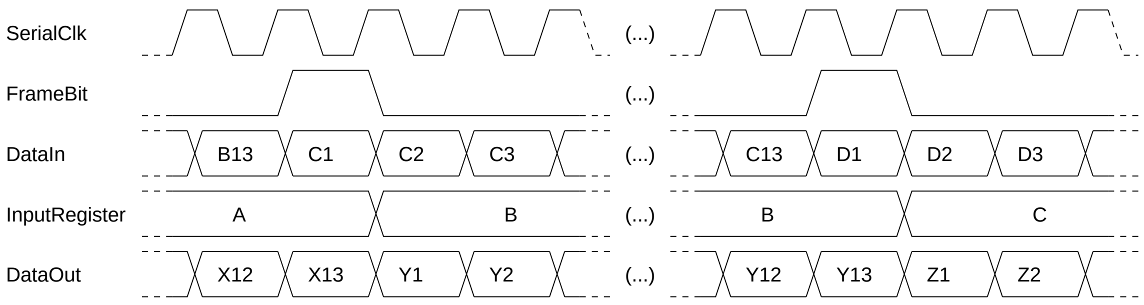


Figure 6.2: Timing diagram of signals relevant for serial interface

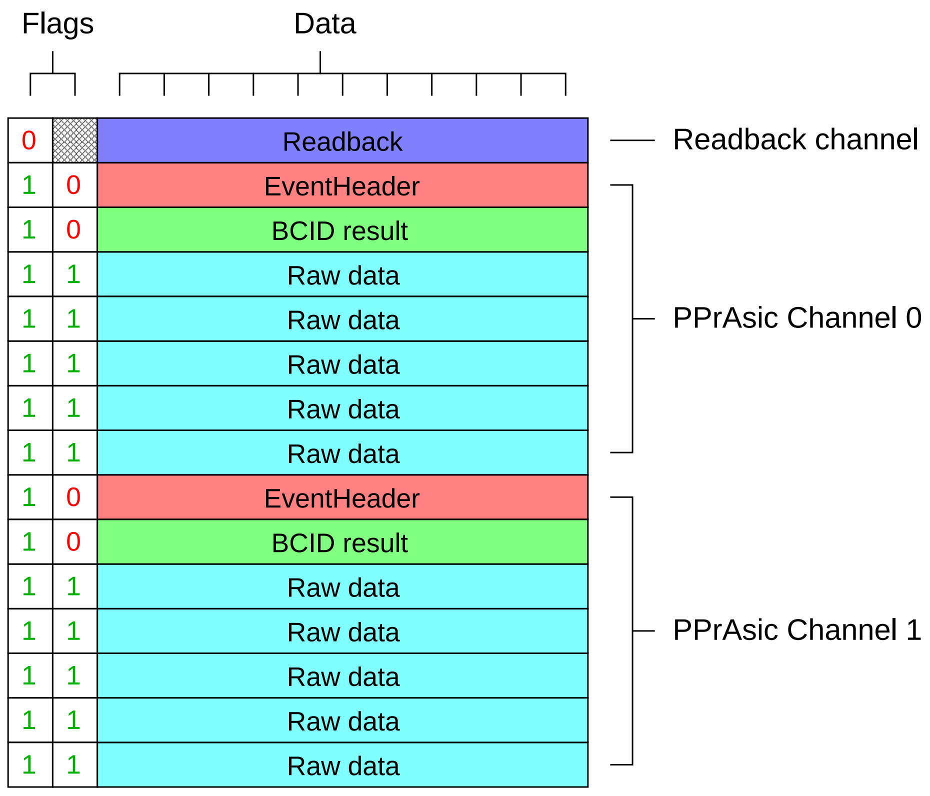


Figure 6.3: Format of readout data

The width of the user data is 11 bits corresponding to the maximum width of internal PPrAsic memories. This results to a serial word length of 13 bits, which has to be indicated by the *Frame* signal.

The readback data stream can contain two different types of information, control and data words. The flag bit not required for channel identification is used to distinguish the two types. This allows to read memory contents with a word width of 11 bits from the PPrAsic in a compact way. See section 4.2 for details.

For configuration, a similar format is be used which contains control and data words, identified by control bits. With this format, it is possible to send 11 bit data words to the PPrAsic in order to load memories efficiently. See section 4.1 for details.

6.3 Bandwidth considerations

If the serial clock is run with 40 MHz and the maximum number of five raw data samples is read out, it takes 4.875 μs to read out an event from one serial interface of the PPrAsic. Even if two serial interfaces are daisy-chained the readout time lies below the maximum of 10 μs, which is the maximum time at a trigger rate of 100 kHz. By reading out fewer raw data the readout time is reduced.

Another way to increase the bandwidth is to use a serial clock with higher frequency. This requires synchronisation of serial clock and the system clock used for the rest of the Asic. Circuitry for this synchronisation is provided on the PPrAsic.

The bandwidth of the readback channel with 40 MHz serial clock and maximum number of raw data samples is 2.2 MBit/s without daisy-chaining. For example the readback of the one histogramming memory (256 * 11 Bit) would require 1.25 ms.

Chapter 7

Frequency Detection

The PPrAsic includes a frequency detection circuit, which is able to discriminate a frequency applied at a dedicted input pin. This frequency detection is forseen to be used for monitoring of the Phos4 phase detection output, which sends a signal with a frequency between 2 MHz and 6 MHz, when the Phos4 chip is operating correctly.

The PPrAsic frequency detector checks if the signal applied at the *FreqIn* input has a frequency higher than 0.3 MHz. If the frequency is lower or the signal has a constant value, the *FreqLost* output is set to 1. The frequency detection works reliably for frequencies up to 20 MHz. Higher frequencies may be not detected correctly, depending on how phase and frequency are related to the 40 MHz system clock of the PPrAsic.

The *FreqLost* signal is also included in the readback status word described in section 4.2.

Chapter 8

JTAG interface

The PreProcessor ASIC is equipped with Boundary Scan logic, surrounding the internal logic of the chip, for board level test purposes according to the IEEE testing standard 1149.1-compliant (or compatible). It is customary to refer to this standard as the ”JTAG” proposal. ”JTAG” is an acronym for the Joint Test Action Group.

8.1 Test access port

Five I/O ports/pins on the PPrASIC chip, exclusively dedicated to Boundary-Scan (this is required by the standard), comprise the Test Access Port (TAP). The JTAG TAP signals, and the actual port names on the PPrASIC are shown in table 8.1.

The communication with the on-chip Boundary-Scan logic is controlled, according to a standard protocol, through JtagTrst, JtagTck and JtagTms ports. Test data is serially shifted into and out of the chip through the JtagTdi and JtagTdo ports respectively.

8.2 Boundary Scan

All the input/output ports (except for the TAP) on the PPrASIC chip are equipped with a Boundary Register cell which are in turn connected serially together between the JtagTdi and JtagTdo ports. This is the so called JTAG Boundary-Register (JTAG BR). The chip also contains the mandatory Bypass Register and an Identification Register. Some user defined Data Registers are also implemented on the chip.

Signal Name	Port Name
Test Reset (TRST*)	JtagTrst
Test Clock (TCK)	JtagTck
Test Mode Select (TMS)	JtagTms
Test Data In (TDI)	JtagTdi
Test Data Out (TDO)	JtagTdo

Table 8.1: List of JTAG signals/ports

8.3 Internal Scan

The Internal Scan logic is implemented as a number of scan chains each connected between the JtagTdi and JtagTdo ports as Data Registers. Each Data Register (JTAG DR) is referenced with a unique Instruction number.

The internal scan paths include the state registers of all state machines and registers used to store intermediate results of processing, which are not accessible by readout or readback logic.

Chapter 9

Registers

9.1 Accessing Registers

The registers of the PPrAsic are accessible via the serial interface. All registers can be written and read back. Each trigger tower channel of the PPrAsic has a set of registers, which are identical from channel to channel. There is also a small global register block, which is used by all blocks controlled by one serial interface, i.e. two for the 4-channel PPrAsic.

The common protocol for accessing internal blocks of the PPrAsic by the serial interface is described in chapter 4. For registers the memory bit of the *CfgAddress* has to be cleared. The global address selects the register block of the channels or the global block. The local address selects the register to be accessed. Figure 9.2 shows a list of all registers of a channel and figure 9.1 shows a list of all global registers.

9.2 Global registers

9.2.1 Readout and Readback

ReadMainEnable If set to 1, readback is enabled. Readout is only enabled, if also register *ReadoutEnable* is set. (default: enabled)

ReadoutEnable If set to 1, readout is enabled. This only takes effect, when also register *ReadoutMainEnable* is set. (default: disabled)

Num	Bits	Name	Def.	Description
0	1	ReadMainEnable	0	Main enable for readout
	2-3	ReadoutEnable	0	
1	0-1	ReadoutSyncRaw	0	Enable external start of raw data readout
	2-3	ReadoutSyncBcid	0	Enable external start of bcid result readout
2	0-4	GenOut	0	Generic outputs
3	0-1	JetSumMode	0	Jet summing mode
	2	BypassBcMux	0	Bypass control for BC multiplexing
	3	ChannelSelect	0	Select output channel in BCMux bypass mode
4	0	DaisyChainEnable	0	Enable internal daisy-chaining

Table 9.1: List of global registers

Num	Bits	Name	Def.	Description
0	0	InBcidNegedge	0	Latch external BCID bit with negative clock edge
	1	InDataNegedge	0	Latch ADC data with negative clock edge
	2	ReadoutSource	0	Source of readout data
	3	BypassLut	1	Bypass the LUT
	4	InvMsbDisable	0	Disable inversion of MSB of ADC data
	5	ExtBcidEdgeEnable	1	Enable edge detection for external BCID bit
1	0-6	PipeDelayBcid	10	Delay of pipeline memory for BCID result data
2	0-6	PipeDelayRaw	10	Delay of pipeline memory for raw readout data
3	0-3	SyncDelayBcid	0	Input delay of external BCID bit
	4-7	SyncDelayData	0	Input delay of ADC data
	8	SyncBypassBcid	1	Bypass synchronisation FIFO for external BCID signal
	9	SyncBypassData	1	Bypass synchronisation FIFO for ADC data
4	0-3	FIRCoeff1	0	FIR filter coefficient 1
	4-7	FIRCoeff2	0	FIR filter coefficient 2
5	0-3	FIRCoeff3	1	FIR filter coefficient 3
	4-7	FIRCoeff4	0	FIR filter coefficient 4
6	0-3	FIRCoeff5	0	FIR filter coefficient 5
7	0-9	SatHigh	500	Upper threshold for BCID of saturated pulses
8	0-9	SatLow	50	Lower threshold for BCID of saturated pulses
9	0-9	SatLevel	1023	Energy level detected as saturation
10	0-10	EnergylevelLow	200	Lower energy threshold for BCID decision logic
11	0-10	EnergylevelHigh	300	Upper energy threshold for BCID decision logic
12	0-7	BcidDecision1	0	BCID decision logic LUT for low energy region
	8	SatOverride1	0	BCID result override flag for low energy region
13	0-7	BcidDecision2	0	BCID decision logic LUT for middle energy region
	8	SatOverride2	0	BCID result override flag for middle energy region
14	0-7	BcidDecision3	0	BCID decision logic LUT for high energy region
	8	SatOverride3	0	BCID result override flag for high energy region
15	0-2	StartBit	0	Start bit where the FIR filter result is clipped
	3	PeakFinderCond	0	Condition for peak in peak finder
	4	DelayExtBcid	0	Delay of external BCID bit in decision logic
16	0-2	NumBcRaw	5	Number of raw data BCs to be read out
	3	NumBcBcid	1	Number of BCID result BCs to be read out
17	0	RateEnable	0	Enable Rate-Metering
	1	RateSource	0	Source (ADC/BCID) for Rate-Metering
	2	HisEnable	0	Enable Histogramming when playback disabled
	2	HisSource	0	Source (ADC/BCID) for Histogramming
	3-4	HisOpMode	1	Histogramming operation mode
18	0-9	RateEtThresh	4	Signal threshold for Rate calculation
19	0-7	RateDelTimeL	20	rate calculation time span (lower byte)
20	0-7	RateDelTimeH	5	rate calculation time span (upper byte)
21	0-9	HisEtThresh	4	Signal threshold for Histogramming
22	0-5	HisLowerBcL	1	Lower bunch number to Histogram (lower 6 bits)
23	0-5	HisLowerBcH	0	Lower bunch number to Histogram (upper 6 bits)
24	0-5	HisUpperBcL	37	Upper bunch number to Histogram (lower 6 bits)
25	0-5	HisUpperBcH	53	Upper bunch number to Histogram (upper 6 bits)
26	0-9	LutPedestal	0	Pedestal of ramp pre-loaded into LUT
27	0-10	LutSlope	256	Slope of ramp pre-loaded into LUT

Table 9.2: List of channel registers (1/2)

Num	Bits	Name	Def.	Description
28	0	PlaybackEnable	0	Enable playback mode
	1	PlaybackSync	0	Enable sync of playback by external input
	2	PlaybackOneshot	0	Enable one-shot mode of playback
29	0-7	PlaybackDelayH	0	Playback delay for multi-shot mode (upper byte)
30	0-7	PlaybackDelayL	0	Playback delay for multi-shot mode (lower byte)
31	0-6	AlmostFullRaw	8	Almost full threshold of raw data derandomizer
32	0-6	AlmostFullBcid	20	Almost full threshold of BCID result derandomizer
33	0-7	SaturationValue	255	Value sent as BCID result in override mode

Table 9.3: List of channel registers (2/2)

ReadoutSyncRaw If set to 1, readout of raw data is started by the external *SyncReadout* signal. (default: disabled)

ReadoutSyncBcid If set to 1, readout of BCID result data is started by the external *SyncReadout* signal. (default: disabled)

9.2.2 Output control

JetSumMode Three different modes of jet summing are available:

- **Trunc9** (JetSumMode = 0): Truncate highest bit of internal 10 bit sum. Bit 9 is dropped. If it is set, the sum is set to the maximum value.
- **Trunc0** (JetSumMode = 1): Truncate lowest bit of internal 10 bit sum.
- **NoTrunc** (JetSumMode = 2): No Truncation. The parity bit `ToJpParity` carries the lowest bit of the internal 10 bit sum. The higest 9 bits are put out on the normal Jet outputs `ToJp`. No parity calculation takes place.

(default: no truncation)

BypassBcMux If set to 1, the BC multiplexing is bypassed and the LUT input data of one channel is directly given out to the *ToCp* output, the *BcMuxFlag* and the *ToCpParity* then contain the two LSBs of the 10 bit word used as LUT input. When the FIR filter is set to bypass mode, the word {`ToCp,BcMuxFlag,ToCpParity`} represents the input value of the ADC inputs. Register *ChannelSelect* selects, which channel is given out. (default: bypass BC multiplexing)

ChannelSelect Selects channel to give out in BC multiplexing bypass mode. A value of 0 selects channel 0, a value of 1 selects channel 1. (default: channel 0)

9.2.3 Generic outputs

GenOut These register is directly connected to output pads of the chip. This will be used on the Pre-Processor MCM to control the Sync-Pattern and the Power-Down signals of the LVDS serializer chips. (default: 3)

9.2.4 Serial Interfaces

DaisyChainEnable If these register is set, the two serial interfaces on the PPrAsic are internally daisy-chained. This global register only exists at the first serial interface. Note that before this register is set, the interfaces are not yet daisy-chained. You either have to adjust the serial frame bit or send the commands setting this register twice. (default: no daisy-chaining)

9.3 Channel registers

9.3.1 Input

InBcidNegedge Latch external BCID bit input with negative clock edge, when register set to 1, latch with positive edge, when register set to 0 (default: positive edge).

InDataNegedge Latch ADC data input with negative clock edge, when register set to 1, latch with positive edge, when register set to 0 (default: positive edge).

InvMsbDisable Inversion of the MSB of the ADC data input is disabled, when this register is set to 1. If the register is set to 0, the MSB is inverted. (default: Invert MSB)

SyncDelayBcid Number of clock ticks the external BCID bit is delayed after input. The value written to the register has to be incremented by one to get the effective number of bunch crossing by which the data is delayed, e.g. a value of 0 delays by 1 clock cycle, a value of 5 delays by 6 cycles. To get no delay you have to set the register *SyncBypassBcid* (default: 0, i.e. 1 clock cycle delay).

SyncDelayBcid Number of clock ticks the ADC data is delayed after input. The value written to the register has to be incremented by one to get the effective number of bunch crossing by which the data is delayed, e.g. a value of 0 delays by 1 clock cycle, a value of 5 delays by 6 cycles. To get no delay you have to set the register *SyncBypassData*(default: 0, i.e. 1 clock cycle delay).

SyncBypassBcid If this register is set to 1, the synchronisation FIFO for the external BCID input is bypassed, that means the signal is not delayed at all. If this register is set to 0 the delay set in register *SyncDelayBcid* takes effect. (default: bypass synchronisation FIFO)

SyncBypassData If this register is set to 1, the synchronisation FIFO for the ADC data input is bypassed, that means the signal is not delayed at all. If this register is set to 0 the delay set in register *SyncDelayData* takes effect. (default: bypass synchronisation FIFO)

ExtBcidEdgeEnable Edge detection for external BCID signal is enabled, when the register is set to 1. That means that if the external bit signal is high for more than one clock cycle only the first clock cycle the bit is propagated as high and the other following clock cycle without level change of the external BCID signal are suppressed. If the register is set to 0, the external BCID signal is propagated without change besides synchronisation. (default: edge detection enabled)

9.3.2 BCID

FIRCoeff1 First FIR filter coefficient, signed value (default: 0).

FIRCoeff2 Second FIR filter coefficient, unsigned value (default: 0).

FIRCoeff3 Third FIR filter coefficient, unsigned value (default: 0).

FIRCoeff4 Forth FIR filter coefficient, unsigned value (default: 0).

FIRCoeff5 Fifth FIR filter coefficient, signed value (default: 0).

SatHigh Upper energy threshold used by the algorithm for saturated pulses (default: 500).

SatLow Lower energy threshold used by the algorithm for saturated pulses (default: 50).

SatLevel Energy level that the algorithm for saturated pulses considers as saturation (default: 1023).

EnergylevelLow Energy threshold that determines the lower limit of the middle-energy range and the upper limit of the low-energy range (default: 200).

EnergylevelHigh Energy threshold that determines the lower limit of the high-energy range and the upper limit of the middle-energy range (default: 300).

BcidDecision1 For the high-energy range this number stores for each combination of the results of the three algorithms the correct answer of the BCID Decision Logic. See section 1.3.1 for an exact definition. (default: 0).

BcidDecision2 For the middle-energy range this number stores for each combination of the results of the three algorithms the correct answer of the BCID Decision Logic. See section 1.3.1 for an exact definition. (default: 0).

BcidDecision3 For the low-energy range this number stores for each combination of the results of the three algorithms the correct answer of the BCID Decision Logic. See section 1.3.1 for an exact definition. (default: 0).

SatOverride1 If this register set to 1 the BCID result is overridden by the value of the register *SaturationValue*, when the BCID decision is taken by the algorithms specified for the *high* energy region. If this register is set to 0, the energy calibration LUT output is taken as BCID result. (default: LUT output)

SatOverride2 If this register set to 1 the BCID result is overridden by the value of the register *SaturationValue*, when the BCID decision is taken by the algorithms specified for the *middle* energy region. If this register is set to 0, the energy calibration LUT output is taken as BCID result. (default: LUT output)

SatOverride3 If this register set to 1 the BCID result is overridden by the value of the register *SaturationValue*, when the BCID decision is taken by the algorithms specified for the *low* energy region. If this register is set to 0, the energy calibration LUT output is taken as BCID result. (default: LUT output)

SaturationValue The value of this register is put out as BCID result, when the *SatOverride* register is set for the energy region, which triggers the BCID decision. (default: 255)

StartBit The 16-bit number coming from the output of the FIR-filter has to be reduced to a 10-bit wide number that can be feed into the Look-Up-Table. The StartBit determines where the block of the 10 bits has its starting point (default: 0).

PeakFinderCond This bit determines which condition is used in the peak-finder.
PeakFinderCond = 0 $\Rightarrow bc(t-1) < bc(t0) \geq bc(t1)$
PeakFinderCond = 1 $\Rightarrow bc(t-1) < bc(t0) > bc(t1)$ (default: 0).

DelayExtBcid Stores the number of bunch crossings that the external BCID-bit has to be delayed in order to syncronize it with the results of the other algorithms.

9.3.3 Histogramming

HisEnable If asserted, i.e. set to 1, enables histogramming if playback is disabled, otherwise this operation is disabled. (default: disabled)

HisSource Raw digitized ADC signals are histogrammed if this register is set to 0, otherwise calibrated BCID results are considered. For histogramming raw data the register *HisOpMode* defines how the raw data are mapped to the histogram memory bins. (default: raw data)

HisOpMode Selects an operation mode for the histogramming, which defines how the data to be histogrammed is mapped to the 256 histogram memory bins. If register *HisSource* is set to 1 (BCID results), *HisOpMode* is ignored and the complete 8 bit wide BCID result is histogrammed. When HisSource is set to 0 (raw data) *HisOpMode* defines the following modes how to get the address of the memory location used for histogramming from the raw data word:

0 Truncate lowest two bits.

1 Truncate lowest and highest bit.

2 Truncate highest two bits.

For the modes truncating high bits, the memory location corresponding to the highest possible address is used for histogramming, if the truncated bits are not zero. (default: truncate the lowest and the highest bit)

HisEtThresh Transverse energy threshold (0.25 GeV steps) above which received signals would be histogrammed.
(default: 4 $\sim$ 00 0000 0100)

HisLowerBcL Lower 6 bits of the bunch crossing number which determines the lower limit of the window on the bunch crossings which would be considered by the histogramming module.
(default: 1 $\sim$ 00 0001)

HisLowerBcH The corresponding upper 6 bits of the bunch crossing number which determines the lower limit of the window on the bunch crossings which would be considered by the histogramming module.
(default: 0 $\sim$ 00 0000)

{ HisLowerBc = 1 $\sim$ 0000 0000 0001 }

HisUpperBcL Lower 6 bits of the bunch crossing number which determines the upper limit of the window on the bunch crossings which would be considered by the histogramming module.
(default: 37 $\sim$ 10 0101)

HisUpperBcH The corresponding upper 6 bits of the bunch crossing number which determines the upper limit of the window on the bunch crossings which would be considered by the histogramming module.
(default: 53 $\sim$ 11 0101)

$$\{ \text{HisUpperBc} = 3429 \sim 1101\ 0110\ 0101 \}$$

9.3.4 Rate-Meter

RateEnable If asserted, i.e. set to 1, enables rate-metering, otherwise this operation is disabled. (default: disabled)

RateSource Raw digitized ADC signals are rate-metered if this register is set to 0, otherwise calibrated BCID results are considered. (default: raw data)

RateEtThresh Transverse energy threshold (in GeV) above which received signals would be counted during the "RateDelTime" time interval (see below) for rate calculation. (default: 4)

RateDelTimeL Lower byte of the time interval during which rate should be calculated. The time interval is given in units of 1024 bunch crossings (25.6 μs). (default: 20)

RateDelTimeH Higher byte of the time interval during which rate should be calculated. The time interval is given in units of 1024 bunch crossings (25.6 μs). One LSB of the `RateDelTimeH` register coresponds to 6.5536 ms. (default: 5)

The combination of the default values for the two `RateDelTime` registers results to a default interval of 33.28 ms.

9.3.5 Readout

PipeDelayBcid Offset of read pointer to write pointer of pipeline memory for BCID results (default: 10)

PipeDelayRaw Offset of read pointer to write pointer of pipeline memory for raw data (default: 10)

NumBcRaw Number of raw data bunch-crossings to be read out. This register may not be changed while readout is active. (default: 5)

NumBcBcid Number of BCID result bunch-crossings to be read out. This register may not be changed while readout is active. (default: 1)

AlmostFullRaw Threshold for setting the almost full flag of the raw data derandomizer. If the number of available memory locations in the derandomizer is less or equal this threshold, the almost full signal is set. (default: 8)

AlmostFullBcid Threshold for setting the almost full flag of the BCID result derandomizer. If the number of available memory locations in the derandomizer is less or equal this threshold, the almost full signal is set. (default: 20)

ReadoutSource This register selects the source for the raw data readout pipeline. If set to 0, the raw data after coarse delay synchronisation as described in section 5.2 is taken, if the register is set to 1, unsynchronized data directly taken from the FADC inputs is fed to the raw data readout pipelines. (default: read out synchronized data)

9.3.6 Playback

PlaybackEnable If set to 1, the playback mode is enabled and input to the processing pipeline comes from playback memory rather than the ADC and external BCID inputs. (default: playback disabled)

PlaybackSync If set to 1, the start of the playback is caused by the external signal *SyncPlayback*. (default: disabled)

PlaybackOneshot If set to 1, the playback mode stops, when the playback memory has been read once. If set to 0, the playback mode continues and the playback memory is read again and again. Between two reads of the memory there is a delay, which can be set with the *PlaybackDelay* registers. (default: continuous mode)

PlaybackDelayH Upper byte of the value of the delay between two playback memory read cycles in continuous playback mode. (default: 0)

PlaybackDelayL Lower byte of the value of the delay between two playback memory read cycles in continuous playback mode. (default: 0)

9.3.7 LUT

BypassLut If set to 1, the LUT table is bypassed, if set to 0, the LUT is used to generate the output of BCID. (default: bypass)

LutPedestal Pedestal of ramp loaded into LUT on reception of the LoadLUT command. (default: 0)

LutSlope Slope of ramp loaded into LUT on reception of the LoadLUT command. (default: 256)

Chapter 10

Pin Description

Table 10.4 gives an overview of the function of the signals. For detailed description see the corresponding chapters in this document.

The pad and pin identifier will be assigned when the final layout and bonding diagram are completed.

Figure 10.1 shows the positions of the pads on the die.

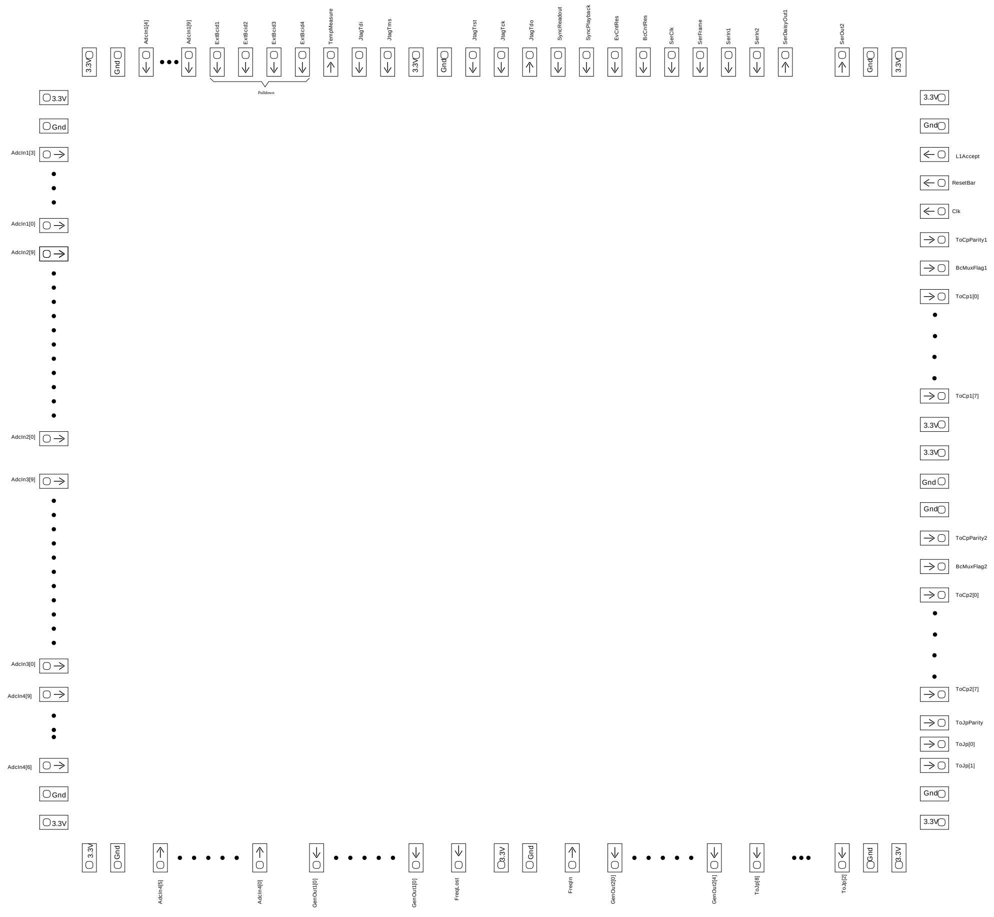


Figure 10.1: Layout of PPrAsic pads

Pad	Pin	Signal	Pad Type
	6	AdcIn1_0	IB15P
	5	AdcIn1_1	IB15P
	4	AdcIn1_2	IB15P
	3	AdcIn1_3	IB15P
	127	AdcIn1_4	IB15P
	126	AdcIn1_5	IB15P
	125	AdcIn1_6	IB15P
	124	AdcIn1_7	IB15P
	123	AdcIn1_8	IB15P
	122	AdcIn1_9	IB15P
	16	AdcIn2_0	IB15P
	15	AdcIn2_1	IB15P
	14	AdcIn2_2	IB15P
	13	AdcIn2_3	IB15P
	12	AdcIn2_4	IB15P
	11	AdcIn2_5	IB15P
	10	AdcIn2_6	IB15P
	9	AdcIn2_7	IB15P
	8	AdcIn2_8	IB15P
	7	AdcIn2_9	IB15P
	26	AdcIn3_0	IB15P
	25	AdcIn3_1	IB15P
	24	AdcIn3_2	IB15P
	23	AdcIn3_3	IB15P
	22	AdcIn3_4	IB15P
	21	AdcIn3_5	IB15P
	20	AdcIn3_6	IB15P
	19	AdcIn3_7	IB15P
	18	AdcIn3_8	IB15P
	17	AdcIn3_9	IB15P
	40	AdcIn4_0	IB15P
	39	AdcIn4_1	IB15P
	38	AdcIn4_2	IB15P
	37	AdcIn4_3	IB15P
	36	AdcIn4_4	IB15P
	35	AdcIn4_5	IB15P
	30	AdcIn4_6	IB15P
	29	AdcIn4_7	IB15P
	28	AdcIn4_8	IB15P
	27	AdcIn4_9	IB15P
	121	ExtBcid1	IB55P
	122	ExtBcid2	IB55P
	123	ExtBcid3	IB55P
	124	ExtBcid4	IB55P

Table 10.1: Assignment of pins, pads and signals (1/3)

Pad	Pin	Signal	Pad Type
	90	ToCp1_0	OB33P
	89	ToCp1_1	OB33P
	88	ToCp1_2	OB33P
	87	ToCp1_3	OB33P
	86	ToCp1_4	OB33P
	85	ToCp1_5	OB33P
	84	ToCp1_6	OB33P
	83	ToCp1_7	OB33P
	76	ToCp2_0	OB33P
	75	ToCp2_1	OB33P
	74	ToCp2_2	OB33P
	73	ToCp2_3	OB33P
	72	ToCp2_4	OB33P
	71	ToCp2_5	OB33P
	70	ToCp2_6	OB33P
	69	ToCp2_7	OB33P
	92	ToCpParity1	OB33P
	78	ToCpParity2	OB33P
	91	BcMuxFlag1	OB33P
	77	BcMuxFlag2	OB33P
	67	ToJp_0	OB33P
	66	ToJp_1	OB33P
	61	ToJp_2	OB33P
	60	ToJp_3	OB33P
	59	ToJp_4	OB33P
	58	ToJp_5	OB33P
	57	ToJp_6	OB33P
	56	ToJp_7	OB33P
	55	ToJp_8	OB33P
	68	ToJpParity	OB33P
	105	SerClk	IB15P
	104	SerFrame	IB15P
	103	SerIn1	IB15P
	102	SerIn2	IB15P
	101	SerOut1	OB33P
	100	SerOut2	OB33P
	95	L1Accept	IB15P
	106	BcCntRes	IB15P
	107	EvCntRes	IB15P
	108	SyncPlayback	IB15P
	109	SyncReadout	IB15P
	93	Clk	IB15P
	94	ResetBar	IB15P
	117	TempMeasure	IOA5P
	46	FreqLost	OB33P
	49	FreqIn	IB15P

Table 10.2: Assignment of pins, pads and signals (2/3)

Pad	Pin	Signal	Pad Type
	116	JtagTdi	IB35P pull-up
	115	JtagTms	IB35P pull-up
	112	JtagTrst	IB35P pull-up
	111	JtagTck	IB15P
	110	JtagTdo	OB93P tri-state
DGND	48,62,65,79,80,96	gnd (I/O)	PP05
DVCC	1,32,33,98,114,129	pwr (Core)	PP04
DGND	2,31,34,99,113,128	gnd (Core)	PP03
DVCC	47,63,64,81,82,97	pwr (I/O)	PP06
	41	GenOut1_0	OB33P
	42	GenOut1_1	OB33P
	43	GenOut1_2	OB33P
	44	GenOut1_3	OB33P
	45	GenOut1_4	OB33P
	50	GenOut2_0	OB33P
	51	GenOut2_1	OB33P
	52	GenOut2_2	OB33P
	53	GenOut2_3	OB33P
	54	GenOut2_4	OB33P

Table 10.3: Assignment of pins, pads and signals (3/3)

Name	I/O	Number	Description
Data input			
AdcIn	I	4 x 10	Input from ADCs
ExtBcid	I	4 x 1	External BCID bit input
Data output			
ToCp	O	2 x 8	To Cluster Processor
ToCpParity	O	2 x 1	Parity for CP output
BcMuxFlag	O	2 x 1	BC Mux flag for CP output
ToJp	O	9	To Jet Processor
ToJpParity	O	1	Parity for JP output
Serial interface			
SerClk	I	1	Serial clock
SerFrame	I	1	Frame bit
SerIn	I	2 x 1	Serial input
SerOut	O	2 x 1	Serial Output
TTC			
L1Accept	I	1	Level-1 Accept
BcCntRes	I	1	BC counter reset
EvCntRes	I	1	Event counter reset
Synchronisation			
SyncPlayback	I	1	Synchronous start of playback
SyncReadout	I	1	Synchronous start of readout
Control			
Clk	I	1	System Clock
ResetBar	I	1	System Reset (active-low)
GenOut	O	10	Generic outputs
Frequency detection			
FreqIn	I	1	Frequency detector input
FreqLost	O	1	Frequency detector result
Temperature sensor			
TempMeasure	O	1	Analog temperature measurement
JTAG interface			
JtagTdi	I	1	Test Data In
JtagTms	I	1	Test Mode Select
JtagTrst	I	1	Test Reset
JtagTck	I	1	Test Clock
JtagTdo	O	1	Test Data Out
Power			
vdd	I	6	Power
gnd	I	6	Ground
Total		117	

Table 10.4: Pins of the PPrAsic

Chapter 11

Technical Data

The PPrAsic is a pure digital design using standard cell logic. As design entry the HDL Verilog is used. The chip incorporates memory cell supplied by the manufacturer. It will be manufactured using the 0.6 μm CMOS process of AMS.

The final layout will approximately consume an area of 65 mm^2. It uses 34000 standard cells and 8.3 kByte of memory, contained in 24 cells.

The core logic and I/O cells use a 3.3 V power supply voltage. Assuming an average power consumption of 4 μW/MHz per standard cell and a signal toggle rate of 40%, the power consumption of the complete chip can roughly be estimated to be 2.3 W.

Appendix A

Internal Protocols

A.1 Configuration and Readback

Internally for configuration and readback a set of signals is used whose names all start with `Cfg`. These are the signals `CfgAddress`, `CfgDataIn`, `CfgDataOut`, `CfgWrite`, `CfgRead` and `CfgDataLast`.

The `CfgAddress` signal represents the address set by a *CfgAddress* or *StartReadback* command (see section 4). It addresses the component, which is subject for configuration or readback. For configuration this can also be two components from different channels. `CfgDataIn` and `CfgWrite` are used by configuration, `CfgDataOut` and `CfgRead` by readback.

A.1.1 Configuration

The `CfgWrite` signal is used as strobe signal. When it is active (set to 1), it indicates that `CfgDataIn` is a valid input data to the component to be configured. It is activated for one clock cycle per data word. If a component cannot process the data within the clock cycle it detects the `CfgWrite` to be activated, it has to store the data provided by `CfgDataIn`. This data is for only one clock cycle guaranteed to be valid.

A.1.2 Readback

Readback is initiated by sending the command *StartReadback* to the PPrAsic. Then internally the `CfgAddress` is set and a sequence of `CfgRead` signals is generated. The component which is read back terminates this sequence by activating the `CfgDataLast` signal.

The `CfgRead` signal is activated (set to 1) for one clock cycle to read back one word. The component address by the `CfgAddress` has to provide a word of readback data on the `CfgDataOut` signal not later than four clock cycles after reception of the `CfgRead` signal. The data has to be valid until the next `CfgRead` signal is received. If the data read back is the last word of a block of data, the `CfgDataLast` signal has to be activated (set to 1), while this data word is present on the `CfgDataOut` signal.

Appendix B

Global Verilog Include File

This appendix reproduces the global PPrAsic include file, which contains most of the definitions regarding word widths, bit locations and order etc. It might be helpful, if the textual information in this manual is not accurate enough.

```
// $Id: incglobal.tex,v 1.2 2000/04/14 17:01:06 huebner Exp $
//
// Global includes
//

// Widths of words propagating through processing pipeline
  `define ADC_WIDTH 10  // Width of input data from FADC
  `define RAW_WIDTH  `ADC_WIDTH+1  // Width of raw data (Adc + ExtBcid)
  `define SYNC_WIDTH `ADC_WIDTH+1  // Width of synchronisation FIFO
  `define BCIDIN_WIDTH `ADC_WIDTH  // Width of data input for BCID logic
  `define BCIDOUT_WIDTH 8  // Width of data output from BCID logic
  `define TOCP_WIDTH 8  // Width of data put out to cluster processor
  `define TOJP_WIDTH 9  // Width of data put out to cluster processor
  `define JETPRESUM_WIDTH 9  // Width of 0.1x0.2 sum
  `define JETSUM_WIDTH 10  // Width of 0.2x0.2 sum
  `define FILTERIN_WIDTH 10  // Width of filter input
  `define FILTEROUT_WIDTH 16  // Width of filter output
  `define LUTIN_WIDTH 10  // Width of Look-up-table address
  `define LUTOUT_WIDTH 8  // Width of Look-up-tabel data
  `define LUTDATA_WIDTH `LUTOUT_WIDTH  // Width of LUT data input bus
  `define PEAKFINDERIN_WIDTH 16  // Width of data put in to peak finder
  `define SATBCIDIN_WIDTH 10  // Width of data put in to saturated BCID logic

  `define SATBCIDDELAY_WIDTH 4 // Width of the SatBcid shift register
  `define EXTBCIDDELAY_WIDTH 8 // Width of the ExtBcid shift register
  // Width of accept bits used as input to decision logic
  `define BCIDACCEPTBITS_WIDTH 3

  `define FIRCOEFF_WIDTH 4 // Width of the FIR-Filter coefficients

  `define SYNCDELAY_WIDTH 4  // Width of synchronisation delay
```

```
// Synchronisation FIFO
  ‘define SYNC_DEPTH 16

// Playback memory
  ‘define PLAYBACKADDR_WIDTH 8  // Number of playback memory address lines
  ‘define PLAYBACKDATA_WIDTH 11   // Width of playback memory
  ‘define PLAYBACKDELAY_WIDTH 16  // Delay after each pipline cycle

  ‘define PLAYBACKMEM_LATENCY 4  // responese time of playback memory

// TTC numbers
  ‘define BCNUMLONG_WIDTH 12  // Bunch crossing number
  ‘define BCNUM_WIDTH 4  // Bunch crossing number
  ‘define EVNUM_WIDTH 4  // Event number

// Pipeline memories
  ‘define PIPEADDR_WIDTH 7  // Width of readout pipeline memory address
  ‘define PIPEDATA_WIDTH 11 // Width of readout pipeline
  ‘define PIPERAW_WIDTH 11  // Width of readout pipeline for raw data
  ‘define PIPEBCID_WIDTH 11 // Width of readout pipeline for bcid results

// Derandomizers
  ‘define DERADATA_WIDTH 11  // Common width of derandomizers
  ‘define DERARAW_WIDTH 11  // Width of derandomizer for raw data
  ‘define DERABCID_WIDTH 11 // Width of derandomizer for BCID results
  ‘define DERAADDR_WIDTH 7  // Common address width
  ‘define DERARAWADDR_WIDTH 7  // Address width of derandomizer for raw data
  ‘define DERABCIDADDR_WIDTH 7  // Address width of derandomizer for bcid data


  // Width of event counter of derandomizers
  ‘define DERAEVCOUNT_WIDTH (‘DERAADDR_WIDTH + 1)

  // This register width has to be the same as the width of the derandomizer,
  // because it is used in a comparation with the addresses of the derandomizer.
  // It is defined here extra because there are two different almost full
  // registers, one for the raw and the other one for the bcid-data, but there
  // is just one module that is used two times for these different cases - this
  // is the PaReadoutDerandomizerFlag inside of the PaReadoutDerandomizer.
  ‘define REG_ALMOSTFULL_WIDTH 7

// Look-up table loading
  ‘define LUTLOADVALUE_WIDTH 19

// Readout

  ‘define READOUTSYNC_1 0  // Bit assignment of register ReadoutEnable
  ‘define READOUTSYNC_2 1  // Each channel gets a bit.

// Serial Interface
```

```
‘define SERWORD_WIDTH 13  // Length of serial frame

‘define SI_LONGFRAME_WIDTH 4  // Length of long serial frame

‘define SI_DATABIT 12   // Data flag bit (B)

// Ctrl word
// Format: B AAAA AAAA ACCC (B=0)
  ‘define SI_ARG_WIDTH 9  // Ctrl word argument width (A)
  ‘define SI_CMD_WIDTH 3  // Ctrl word command width (C)

  ‘define SI_CMD_L 0
  ‘define SI_ARG_L ‘SI_CMD_WIDTH

  ‘define SI_CMD_H ‘SI_CMD_WIDTH - 1 + ‘SI_CMD_L
  ‘define SI_ARG_H ‘SI_ARG_WIDTH - 1 + ‘SI_ARG_L

  // Command word definitions

  ‘define SI_CMDW_NOP           ’b000 // No operation
  ‘define SI_CMDW_CFGADDR       ’b001 // Config target address
  ‘define SI_CMDW_LOADLUT       ’b010 // Pre-load LUT memory
  ‘define SI_CMDW_STARTREADBACK ’b011 // Start of readback
  ‘define SI_CMDW_PLAYBACKRESET ’b100 // Reset of playback memory address
  ‘define SI_CMDW_STOPPIPELINE  ’b101 // Stop of readout pipelines
  ‘define SI_CMDW_READOUTRESET  ’b110 // Reset of readout logic and memories

  // Command argument definitions

  ‘define SI_CMDA_STOPPIPE_H  4 // Bit to select stop of pipleine 1
  ‘define SI_CMDA_STOPPIPE_L  1 // Bit to select stop of pipeline 2

  ‘define STOPPIPEWORD_WIDTH (‘SI_CMDA_STOPPIPE_H - ‘SI_CMDA_STOPPIPE_L + 1)

  ‘define SI_CMDA_STOPPIPE_RAW_1   1 // Stop raw data pipeline channel 1
  ‘define SI_CMDA_STOPPIPE_BCID_1  2 // Stop bcid data pipeline channel 1
  ‘define SI_CMDA_STOPPIPE_RAW_2   3 // Stop raw data pipeline channel 2
  ‘define SI_CMDA_STOPPIPE_BCID_2  4 // Stop rbcid data pipeline channel 2

  ‘define SI_CMDA_READOUTRESET_1   1 // Readout reset channel 1
  ‘define SI_CMDA_READOUTRESET_2   2 // Readout reset channel 2

  ‘define SI_CMDA_PLAYBACKRESET_1  1 // Readout reset channel 1
  ‘define SI_CMDA_PLAYBACKRESET_2  2 // Readout reset channel 2

// Data input word
// Format: B DDDD DDDD DDDD (B=1)

  // Memory configuration
  // Format: 1 TDDD DDDD DDDD
  `define SI_DATA_MEM_TYPE 11  // (T) Type of data: 0: Memory address,
                               //                   1: Memory data

`define READBACK_WIDTH    12  // Width of readback word

// Readback word
// Format : 0 Bddd dddd dddd
//
// Format Data:       0 1DDD DDDD DDDD
// Format Header:     0 0SxC CCCC CCCC
// Format Header Cfg: 0 00xC CCCC CCCC
// Format Status:     0 01FP SSSS HHRR
  `define READBACK_DATABIT     11 // (B) 0: Header, 1: Data
  `define READBACK_DATA_H      10 // (D)
  `define READBACK_DATA_L       0

  // Header
  `define READBACK_STATUSBIT   10 // (S) 1: Status, 0: Cfg Header

  // Cfg Header
  `define READBACK_UNUSED       9
  `define READBACK_CFGADDR_H    8 // (C)
  `define READBACK_CFGADDR_L    0

  // Status
  `define STATUSWORD_WIDTH          10 // Width of status word for readback

  `define STATUS_FREQLOST             9 // Frequency detetctor output
  `define STATUS_PLAYBACK_ACTIVE      8 // Playback is active (ch 1 or ch2)
  `define STATUS_PIPE_STOPPED_RAW_2   7 // Raw data pipeline stopped (ch 2)
  `define STATUS_PIPE_STOPPED_RAW_1   6 // Raw data pipeline stopped (ch 1)
  `define STATUS_PIPE_STOPPED_BCID_2  5 // Bcid result pipeline stopped (ch 2)
  `define STATUS_PIPE_STOPPED_BCID_1  4 // Bcid result pipeline stopped (ch 1)
  `define STATUS_HISTO_AVAILABLE_2    3 // Histogram available (ch 2)
  `define STATUS_HISTO_AVAILABLE_1    2 // Histogram available (ch 1)
  `define STATUS_RATE_AVAILABLE_2     1 // Rate measurement available (ch 2)
  `define STATUS_RATE_AVAILABLE_1     0 // Rate measurement available (ch 1)

  `define READBACK_STATUS_H  9
  `define READBACK_STATUS_L  0

// Data output word
// Format: 1 TDDD DDDD DDDD

  // Width of readout data from processing channels including type bit
  `define READCHANNEL_WIDTH 12

  // (T) Type of readout data: 0: header,bcid results, 1: raw data
  `define SI_DATA_TYPE 11
  // Readout event header word
  // Format: CLH EEEE BBBB
    ‘define EVENTHEADER_CHANNEL      10  // indicates channel number (C)
    ‘define EVENTHEADER_LOSSFLAG      9  // Flag for loss of events (L)
    ‘define EVENTHEADER_HEADERSONLY   8  // Flag for headers only mode (H)
    ‘define EVENTHEADER_EVNUM_H       7  // Lowest 4 bits of event number (E)
    ‘define EVENTHEADER_EVNUM_L       4
    ‘define EVENTHEADER_BCNUM_H       3  // Lowest 4 bits of BC number (B)
    ‘define EVENTHEADER_BCNUM_L       0

// Register Configuration bus
  ‘define CFGADDR_WIDTH 9   // Width of address of register configuration bus
  ‘define CFGDATA_WIDTH 12  // Width of register configuration bus data
  ‘define CFGDATAOUT_WIDTH 11  // Width of register configuration output bus
  ‘define REGDATA_WIDTH 11 // Width of register readback bus

  // Config address
  // Format: GGLL LLLL M

  // Global address (GG):
  // GG = 00 : Address global blocks, which sit in top of channels
  // GG = 01 : Address only channel 0
  // GG = 10 : Address only channel 1
  // GG = 11 : Address both channels
  ‘define CFGADDR_SELECT_1 ‘CFGADDR_WIDTH-2  // Select channel 1
  ‘define CFGADDR_SELECT_2 ‘CFGADDR_WIDTH-1  // Select channel 2

  // Global address (GG) (register bank)
  ‘define CFGADDR_GLOBAL_L ‘CFGADDR_SELECT_1
  ‘define CFGADDR_GLOBAL_H ‘CFGADDR_SELECT_2
  ‘define CFGADDR_GLOBAL_WIDTH ((‘CFGADDR_GLOBAL_H) - (‘CFGADDR_GLOBAL_L) + 1)

  // Local address (L...L) (register number or memory number)
  ‘define CFGADDR_LOCAL_L 1
  ‘define CFGADDR_LOCAL_H ‘CFGADDR_WIDTH-3
  ‘define CFGADDR_LOCAL_WIDTH ((‘CFGADDR_LOCAL_H) - (‘CFGADDR_LOCAL_L) + 1)

  // Local addresses of memory blocks
  ‘define LOCADDR_LUT 1
  ‘define LOCADDR_PLAYBACK 2
  ‘define LOCADDR_PIPERAW 3
  ‘define LOCADDR_PIPEBCID 4
  // Local address of Ratemeter module
  ‘define LOCADDR_RATE 5

  // Memory select bit (M) 0: address registers, 1: address memories
  ‘define CFGADDR_MEMSELECT 0

  // Histogramming and Ratemetering
  `define HISDELAY_WIDTH 8
  `define RATEDATA_WIDTH 11
  `define RATESCALER_WIDTH 10
  `define RATETIMER_WIDTH 16
  `define RATECOUNT_WIDTH 20
  `define REG_RATEDELTIME_WIDTH `RATETIMER_WIDTH

  `define HISMODE_TRUNCLOW  0
  `define HISMODE_TRUNCMID  1
  `define HISMODE_TRUNCHIGH 2

// Jet processor output

  `define JETSUMMODE_TRUNC9   0  // Truncate bit 9 of four-channel jet sum
  `define JETSUMMODE_TRUNC0   1  // Truncate bit 0 of four-channel jet sum
  `define JETSUMMODE_NOTRUNC  2  // No truncation, use parity bit as data bit

// Frequency discriminator

  `define FREQCOUNT_WIDTH 7 // Width of frequency detection counter
```

Bibliography

[1] RD-12, *Timing, Trigger and Control Systems for LHC Detectors*, CERN/LHCC 97-29

[2] Kambiz Mahboubi, *Histogramming & Rate metering in the PPrASIC*, Internal note, Universität Heidelberg, Institut für Hochenergiephysik,
`http://wwwasic.kip.uni-heidelberg.de/docs`