Note: ATL-COM-LARG-99-011
Version: 3.1
Created: 8 December 1998
Last Modified: 30 March 2000

The ROD Demonstrator for the LArgon Calorimeter

Board Description Document

Document Info

Document Title: The ROD Demonstrator Board for the LArgon Calorimeter

Author: The LArgon ROD working group

C.Billat, J.Colas, N.Dumont, R.Lafaye, A.Masserot, G.Perrot, L.Poggioli, J.Prast (LAPP, Annecy)

W. Efthymiopoulos, D. La Marra, A. Leger (Uninversity of Geneva)

F. Henry-Couannier, B. Repetti (CPPM, Marceille)

J. Fent, L. Kurchaninov, H. Oberlack (MPI, Munich)

W. Cleland, R. Engelmann, D. Lissauer, J.Parsons (BNL, Navis, Pittsburg, Stony-Brook)

File Name: lardemorod_v3r1.doc

URL: http://atlasinfo.cern.ch/Atlas/GROUPS/LIQARGON/ROD/rod-docs.html

Release record

Version	Revision	Date	Comments
1	0	11/12/1998	Creation date, first release.
2	0	19/2/1999	Update after first set of comments.
	1	11/3/99	Update part with connector pin-out and TTC information.
	2	15/3/99	Revise wording, add section with system architecture, new version for the TTC distribution with TTCrx in the board.
	3	11/5/99	Update for the May 99 LArgon week
	4	2/7/99	Update for the July 99 LArgon week • Modifications in the event format • Summary table for PU-MB pinout • Power distribution • Board mechanics
	5	30/9/99	• Various comments for the text (mainly grammar) • Information for the MB↔PU communication and the TTC distribution added • Pinout for the MB-PU connectors defined • Update on the board mechanics and dimensions • New P2/J2 and P3/J3 connector pinout
3	0	11/01/00	Final version corresponding to the board being produced • TTCrx chip mezzanine board • VME addressing schema for the motherboard components • Update transition module pin definitions
	1	30/03/00	Few text modifications to correct minor errors.

Table of Contents

1 Introduction

The goals of the demonstrator project

The main goal of the **ROD demonstrator (RODDemo)** board is to serve as an intermediate step towards the construction of the full ROD module for the ATLAS LArgon calorimeter. It will be the test-bed for the R&D work within the following 2.5 years, where all the functions foreseen for the final ATLAS module could be developed and tested.

In this document a detailed description of the RODDemo board is given. This is a working document, which will be continuously updated as the work advances, therefore the readers have always to pay attention to the date and version number of the document in hand. . More information about the board can be found in the LArgon Web pages.

This document is organised as follows: In the Introduction section a general description of the demonstrator board and its basic components is given. In the following sections the board components are described in detail, followed by the sections on the input/output interfaces connector pinout allocation and output data format. The document ends with a discussion on the points that the demonstrator program will not address as well as the upgrade path towards the final ATLAS board.

1.1 Board Architecture - Requirements

The ROD board receives data from the **Front-End Boards (FEB)** via a set of optical (or copper) links. It does the data processing to evaluate the quantities like energy and time for each channel and outputs data to the ROB boards to be used first for the LVL2 and the DAQ[1]. For the processing task the board has several **DSP Processing Units (PU)** into which the input data are distributed.

The baseline architecture parameters for the ROD board as described in the TDR are shown in **Table 1** along with those foreseen for the ROD Demonstrator.

	ROD Baseline	ROD Demo
Input links (32 bits @ 40 MHz)	2	2
Number of channels per board	256	256
Number of DSP Processing Units	8	4
Number of channels / DSP PU	32	64
Output links (800 Mb/s)	1	1

Table 1 Main characteristics of the ROD Demonstrator board, compared to the baseline readout architecture as described in the TDR.

The choice of RODDemo parameters is based on the following requirements:

Channel density: 64 channels per PU

Recent developments in the DSP processing power indicate that such a target is not far from what can be achieved. If this is proven, it might be that for the final ROD module 8 PU can be used, thus having double the density of that described in the TDR with important implications in the overall design and cost of the system.

Input/Output Links: two input links and one output link

As described in the TDR baseline architecture. For the output, one link per 256 channels is foreseen, since some data reduction is expected at the ROD level.

Test beam: test with the real calorimeter signals

It should be able to test the demonstrator board with the real calorimeter signals in the test beam. This imposes some additional constraints to the board design such as:

- It should be able to analyse all the input channels without rejecting any,
- It should be able to have full readout by VME to be compatible with the DAQ system of the test beam, and
- It should be able to analyse the calibration data as foreseen in ATLAS.

ATLAS tests: test with the LVL2 and DAQ-1 prototypes

It should be able to test the RODDemo with the prototypes of the ROB and DAQ systems when they become available. Tests of the full readout chain of the detector from the FEB up to the EF/DAQ should be possible to be made.

Modularity: evaluate different DSP technologies and board architectures

It is expected that within the following years the DSP technology will evolve quite fast, therefore we should be able to profit the most of it. The same applies to all the peripheral components such as memories, FPGAs, FIFOs used in many places in the board. Therefore a modular design for the board has to be adopted, where the basic components can be easily modified or upgraded.

Full functionality: as in the final module

All the functions foreseen for the final ATLAS module should be implemented in the demonstrator board. In particular the board should:

- issue the appropriate busy signals and do all the dead-time handling,
- issue error messages and monitoring information for its status and operation,
- use the TTC standard signals and information, including the global reset signal when an out-of-sync condition is detected, and
- send an event with the defined ATLAS format for each trigger received.

This list is not exhaustive and is subject to change as the overall design of the system advances.

General constraints: Cost and form factor

It is probable that the cost for the prototype boards will be different than the one quoted for ATLAS. Either because some of the components might be expensive today, in particular those at the cutting-edge of the technology, or because some additional components -eg. additional memory- might be needed to help for the testing and the R&D work. Nevertheless, some effort should be made to keep the cost of the board at a reasonable level.

Also, for the first prototypes the requirements in terms of space might be somehow relaxed, but eventually, it should be shown within the demonstrator program that all the foreseen input/output links and PU boards can be accommodated within a 9U VME board in an optimal and cost efficient way.

1.2 The Demonstrator Board

A block diagram of the demonstrator board is shown in **Figure 1**. It is a 9U VME board, which consists of two parts:

- The **motherboard**, which has all the I/O connections and controls, and the VME interface,
- Four **DSP Processing units (PU)** mounted as mezzanine cards where all the data processing is done.

The main characteristics of the demonstrator board are described in **Table 2**.

Item	
Crate type	VME64x standard, with custom P3 back-plane
Board type	9U
Board dimensions	366.7 × 400 mm

DSP Processing Units	4
Mezzanine board dimensions	85 × 185 mm
Number of channels in the board	256
Number of channels per PU	64
Inputs	
FEB links	2 (32-bits @ 40 MHz + clock)
TTC input	CLCK, BCRST, BCID, EVID, EVCRST, Ttype, Reset…
Outputs	
ROB links	1.28 Gbit/s (32-bits @ 40 MHz)
VME bus	Full readout and board control
BUSY signal	Asynchronous

Table 2 The basic characteristics of the ROD demonstrator board

From the motherboard point of view the PU is seen as FIFOs, where the data are continuously clocked-in using the clock from each link. All the data handling and decoding is done in the PU.

A similar solution is adopted for the output data stream from the PU cards. The **Output Controller** (**OC**) "pulls" the data from the mezzanine board using its own clock, adds the event header and trailer and sends the data either to the output link or to the Output Buffer to be read asynchronously by the VME, or to both.

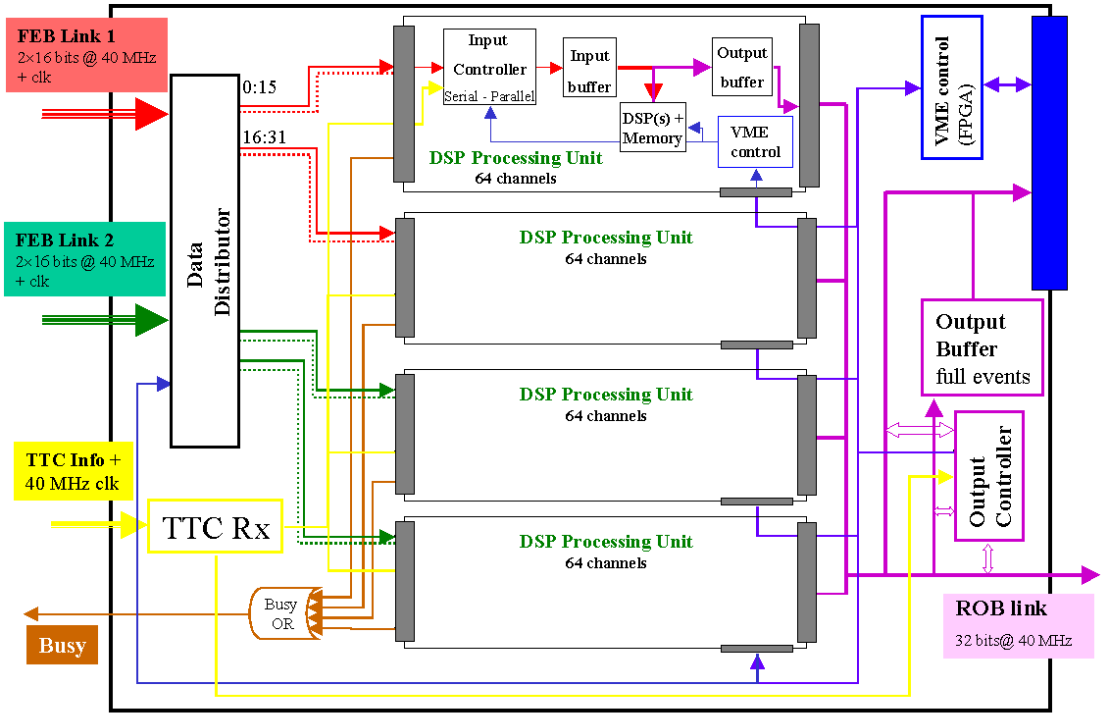


Figure 1 Block diagram of the ROD Demonstrator Board.

The TTC information is received from the back plane and it is treated by the motherboard (using the TTCrx chip), before being distributed to the PU boards and the OC.

In the following sections a detailed description of each of the components is given.

2 The Motherboard

The motherboard is a full size 9U VME module able to carry four mezzanine boards, which span the full board height.

2.1 Mechanics

The board dimensions according to the VME64x standard are:

- Motherboard: H366.7 × W400 × D2.4 (mm)
- Transition module (back of VME crate): H120(or 160) × W400 × D2.4 (mm)
- Processing unit mezzanine: H85 × W185 × D1.6 (mm)

The clearance between the two mezzanines is 4mm, increasing to 7.3mm at each side of the board in order not to interfere with the guiding rails in the crate. **Figure 2** shows the cross-section of the board. The connector height is chosen such that SMD components can be placed on the motherboard and the inside face of the PU as well.

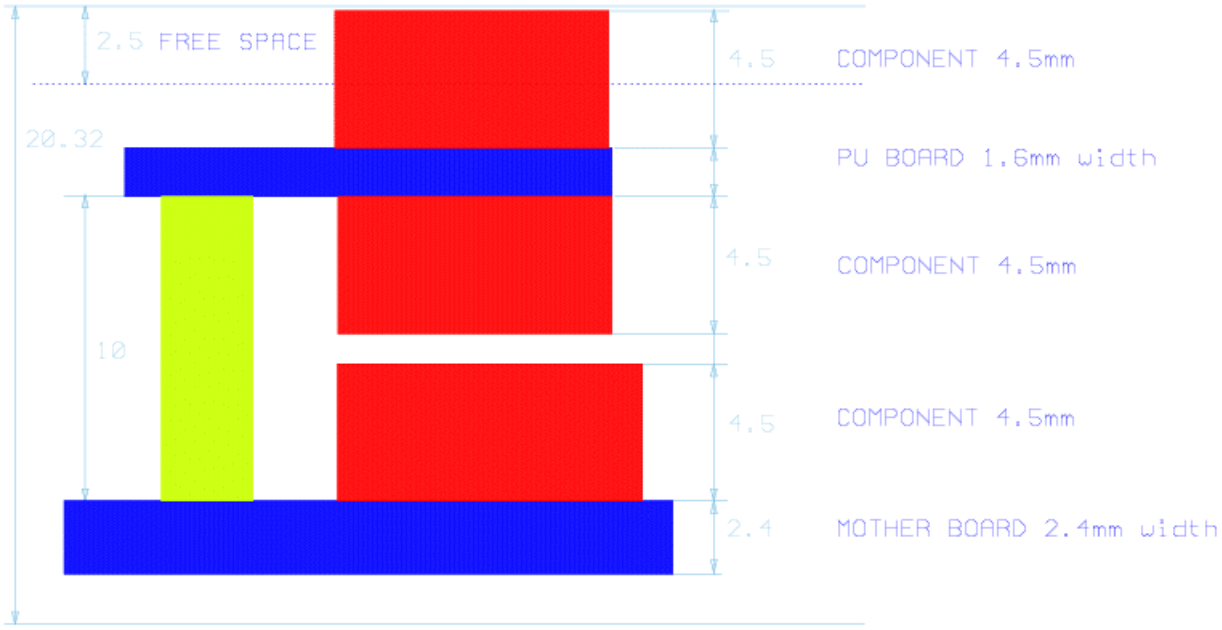


Figure 2 RODDemo board cross-section.

The final board width will depend on the size of the actual components used and it might overpass the allowed single slot width.

The connectors used and their topology is discussed in Section 5.1.

2.2 Input/Output Links

All the I/O links for the card are from the P2 and P3 VME connectors. The input (output) link receiver (transmitter) are housed in a special "Transition Module" which mounts at the back of the VME crate as shown in **Figure 3**.

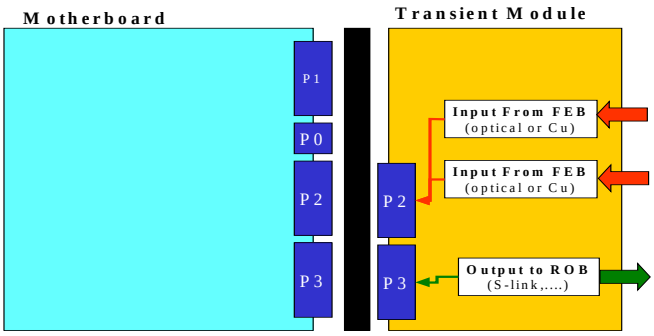


Figure 3 A schematic diagram of the motherboard and the Transition Module used for the input/ouput links.

The actual links can be either optical or from copper cable, but the pinout arrangement to the ROD remains the same. With respect to the ROD, the input (output) data are sent (received) as 3.3V TTL signals via the P2 and P3 connectors. Detailed information for the Transition Module design can be found in [2].

Using the proposed VME64x standard, in each of the P2/J2 & P3/J3 connectors we will have 160 pins, which is considered as sufficient for this case. The exact pin-out diagram for the input and output connectors can be found in Section 5.1.2.

2.3 The TTC Information and BUSY Signal

The TTC and BUSY signal distribution in the ROD crate is done using a special module called **TTC Interface and Busy Module (TBM)** as seen in **Figure 4**. The TBM module receives the TTC signals from the trigger system, does the optical to electrical conversion and then sends them through the VME back plane in all the ROD boards in the crate. Through the back plane it also collects the BUSY signals from all the RODs and provides the crate BUSY as well as dead-time monitoring information to the local CPU.

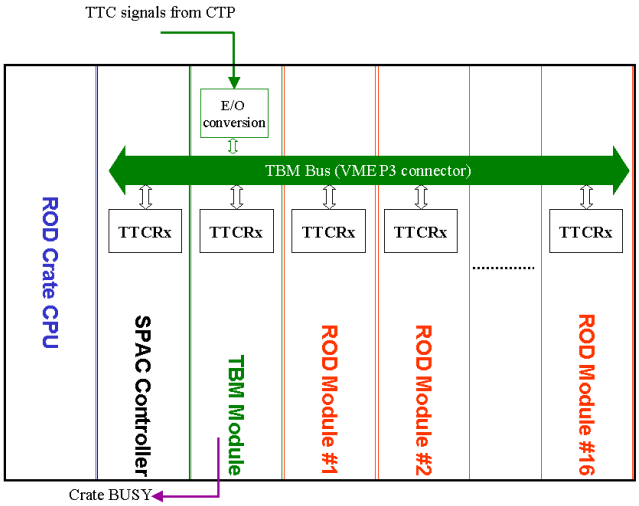


Figure 4 The TTC signal distribution in the ROD crate.

2.3.1 The TBM bus

The TBM bus in the VME back plane consists of the following signals:

- **TTC_B00, TTC_B01**: two lines to send the TTC information. More details for the TTC signals and the TTCrx chip functions can be found in [3].
- **BUSY_B#**: which is the BUSY signal from each ROD module.

The pinout allocation in the P3 VME connector can be found in Section 5.1.3

2.3.2 TTC signal distribution in the ROD

At the input of each ROD the TTC data are recovered by the TTC Controller, which distributes them in the motherboard, and in the PU as schematically shown in **Figure 5**.

Apart from the TTC_clock, the **TTC controller** provides

- The BCID(12-bits) and TriggerType (8-bits) to each PU
- The BCID(12-bits), EventID(24-bits) and Ttype(8-bits) to the Output Controller

A common 8-bit bus is used, and the data are maintained for two clock cycles and the TTC controller provides the necessary write signals: TTC_BCIDWr, TTC_TtypeWr.

In each PU the BCID and Ttype data are stored in separate synchronous FIFOs. To get the full BCID two consecutive readings are needed. The Output Controller needs two FIFOs: one for storing the BCID and EventID, and the other for the Ttype information. To get the BCID again two readings are needed, and three for the EventID, as show in **Figure 6**.

At each L1A receipt, the BCID and EVID information is directly available, while the Ttype information arrives later, without a fixed latency, but the order is maintained. The timing for the TTC signals will be made such that the BCID information is available in the PU before the data from the FEB. The Trigger Type information should be available before the end of the event, so the algorithm in the DSP can use it.

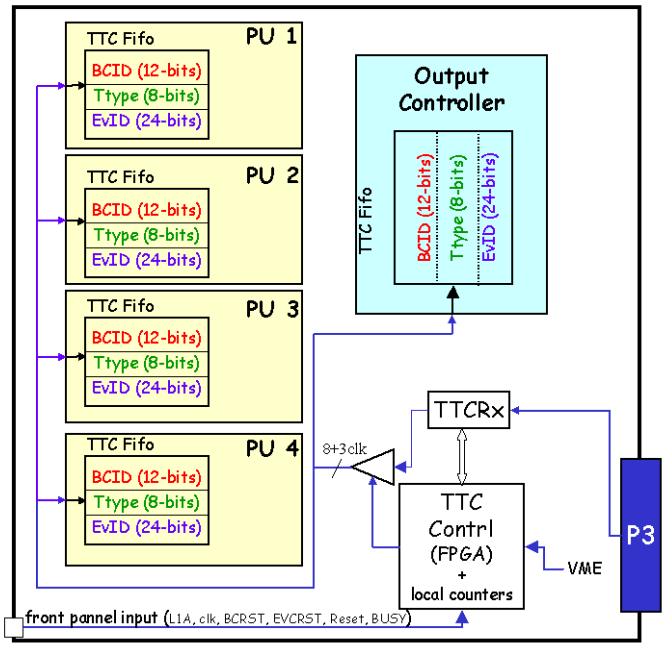


Figure 5 The TTC and BUSY signal distribution into the ROD module.

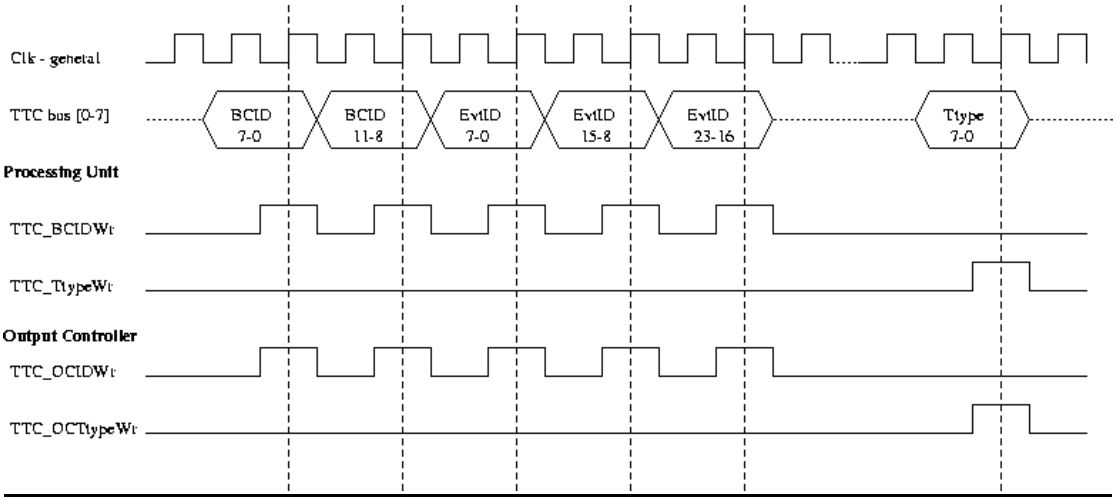


Figure 6 Timing of the TTC distributor signals.

Local mode

This to operate without the TBM signals but taking the necessary input from a front panel connector. The front panel pins are shown in **Table 3**.

Signal	Direction (from ROD)	# Pins	Level
Clock	Input	2	LVDS
L1A	Input	2	LVDS
BCRst	Input	2	LVDS
ECRst	Input	2	LVDS
BUSY	Output	2	LVDS
Reset	Input	2	LVDS

Table 3 Front panel signals.

In this case the BCID and EventID information is generated from internal counters. These counters are reset by the BCRst and ECRst signals. The Ttype information is loaded by VME.

VME mode

There is also the possibility to generate a complete sequence only by VME. In this case the sequence of VME write should be: Trigger-Type, Event-ID, BunchCrossing-ID. The data are send to the PU and the OC only when the BCID is received.

TTCrx mode

This is the configuration for the final board, where a TTCrx chip is mounted in a special mezzanine board on the motherboard which takes the TTC signals from the TBM bus. The schematics of this mezzanine board are shown in **Figure 7**.

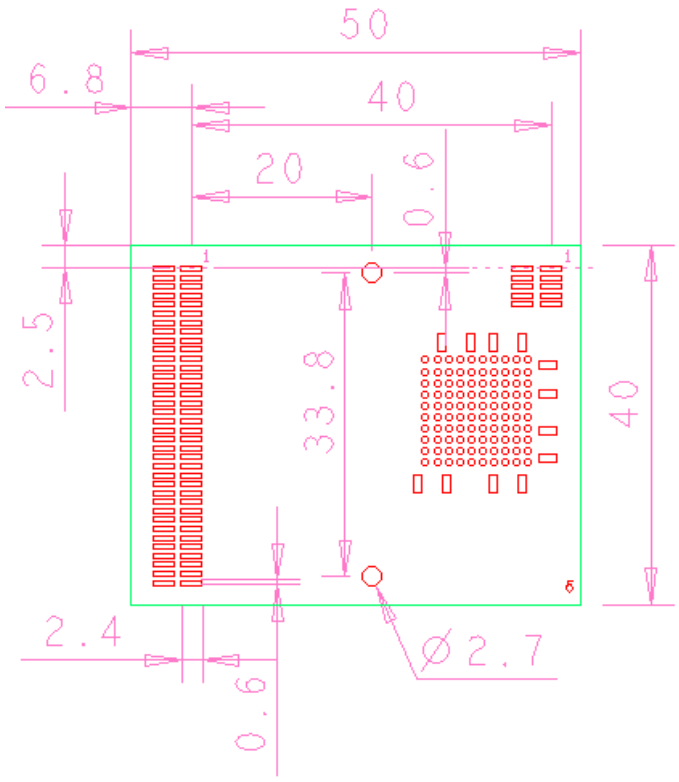


Figure 7 Bottom layer of the TTCrx mezzanine board seen by transparency from the top of the board.

The TTCrx mezzanine board incorporates a TTCrx chip with capacitive coupling on the input. It doesn't use a PROM to configure the chip and therefore must be initialised either by the I2C interface (SDA, SCL) of by the TTC input itself. The differential TTC input signal (in, in_b) can have an amplitude from 20mV up to 1V pp. The TTCrx identification number (ID) is set by resistors connected to GND or VDD. Due to space constraints only 8 of the 14 bits for ID can be set with a dip switch.

- Master mode is set to 00 (= no test mode and serial to parallel conversion on channel B enabled).
- The ID addresses are configured using an 8-bit switch. 256 TTC addresses (bits 3:10) and 8 I2C addresses (bits 2:5) can be configured. This also means that it is possible to have up to 32 TTCrx chips using the same I2C address.
- There is only one power supply which can be 3.3V or 5V.
- The part numbers for the connectors used are: SAMTEC CLM136-02-LD, SAMTEC CLM105-02-LD.

In **Table 4** the signals used by the TTC Controller are listed:

Signals	Description	#Pins
	Input Signals to the TTC Controller	
Configuration	**From VME Interface**	
Config_TTCrx_Local	1=local or VME mode, 0=TTCrx mode.	1
Config_VME	1=VME mode, 0=local mode.	1
Local Mode	**From Front Panel**	
LV1A	LV1 used for BCID and EvtID local counters.	1
BCRst	Reset BCID counter.	1
EvtRst	Reset EvtID counter.	1
Reset	Resynchronisation reset	1

TTCrx Mode	**From TTCrx**	
EvtCntHStr	High to validate the 12 bits MSB of the BCIDon BCnt bus.	1
EvtCntLStr	High to validate 12 bits LSB of the EvtID on BCnt bus.	1
BCntStr	High to validate 12 bits MSB of the EvtID on BCnt bus.	1
BCnt	12 bit data bus.	12
SubAdr	8 bit sub-address bus ('00000000' for Ttype).	8
DoutStr	High to validate sub-address and data on Dout bus.	1
Dout	8 bit data bus for the TTYPE.	8
Brcst	6 bit broadcast bus (used to generate the synchronisation reset)	6
BrcstSTr1	High to validate data on Brcst bus	1
VME Mode	**From VME Interface**	
BCID_VME	BCID 12 bits register written by VME.	12
EvtID_VME	EvtID 24 bits register written by VME.	24
LV1A_VME	BCID and EvtID registers transferred to PU and OC on Clk_General rising edge when high.	1
Dout_VME	TTYPE 8 bits register written by VME.	8
DoutStr_VME	TTYPE registers transferred to PU and OC on Clk_General rising edge when high.	1
Anothers	**Common to the Mother Board**	
Reset	Reset the chip.	1
Clk_General	General clock of the mother board.	1
	Total	
	Output Signals from the TTC Controller	
To P.U		
TTC_BCIDWr	High on Clk_General to write BCID in PU BCID fifo.	1
TTC_TTypeWr	High on Clk_General to write TTYPE in PU TTYPE fifo.	1
To O.C		
TTC_BCIDWrOC	High on Clk_General to write BCID+EvtID in OC BCID+EvtID fifo.	1
TTC_TTypeWrOC	High on Clk_General to write TTYPE in OC TTYPE fifo.	1
To P.U and O.C		
TTC_Data	8 bits data bus for BCID, EvtID and TTYPE.	8
TTC_Rst	To reset the TTC counters (needs to be defined further)	1
	Total	

Table 4 Input signals to the TTC Controller.

2.4 The Data Distributor

The Data Distributor is a multiplexer, which selects between the FEB and VME data stream to send to the PU. The input data are distributed equally with 8 ADCs per PU, as shown in **Table 5**. The virtue of the FEB data transmission is that the data from a single ADC (8 channels) are send using only two lines of the link, therefore 16 lines of data are connected for each PU. The clock and the additional bit with the error flag received by the input link are send as well.

ADC	a, b, c, d, e, f, g, h	I, j, k, l, m, n, o, p
FEB Link #1	PU 1	PU 2
FEB Link #2	PU 3	PU 4
VME	PU1, PU2	PU3, PU4

Table 5 The ADC distribution from the input links into the four PU.

For the case of the VME data injection, the same 32-bits is used for simulating both the FEB links. In this case PU1 and PU2 (PU3 and PU4) will get the same data.

2.5 The Output Controller

The main functions of the Output Controller (**OC**) are listed below:

- Receives and stores the TTC information send from the TTC Controller.
- Reads the event fragments from the four PU output buffers.
- Builds and Formats the complete ROD event fragments.
- Outputs the data either to the VME or to the ROD-ROB link or in both, according to its configuration.
- Outputs periodically or by trigger type the data into the spy buffer, according to its configuration.

Further details for the Output Controller design can be found in [4].

2.5.1 Block diagram

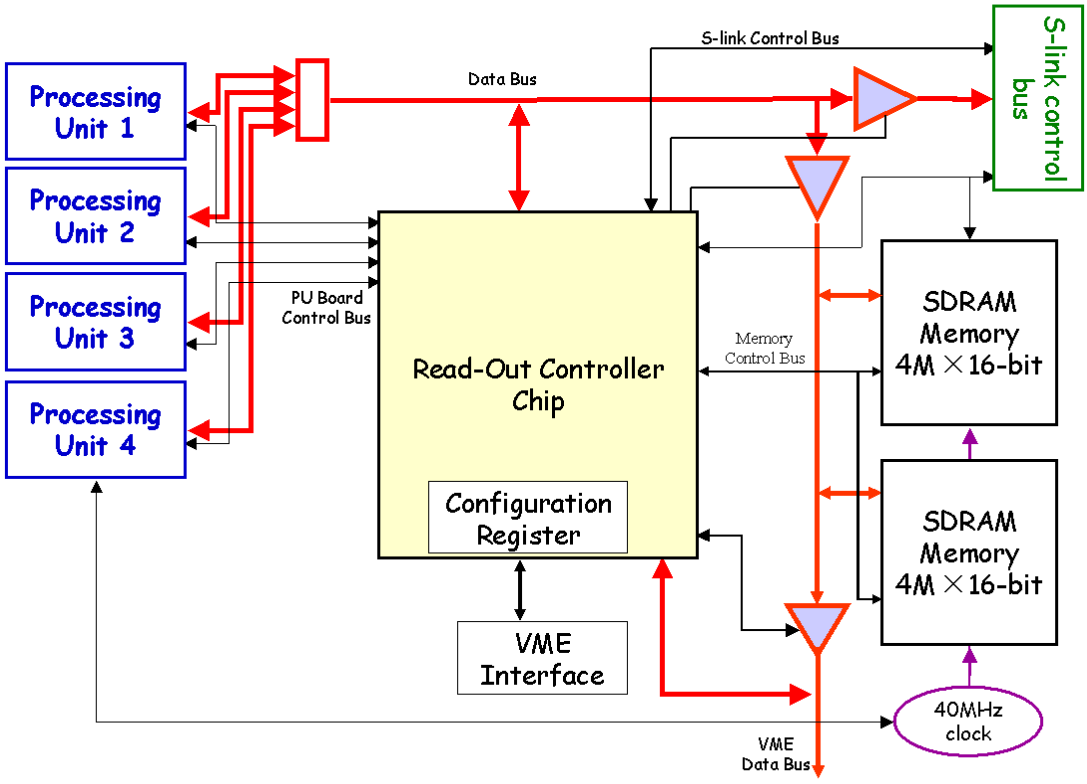


Figure 8 Schematic drawing of the output controller.

The block diagram of the output controller is shown in **Figure 8**. As seen there, it must be able to manage three kinds of control bus:

- the four PU Board Control Bus: a 5-line buts which controls the output of each PU board,
- the S-link Control Bus, which allows the OC to control the data, which is send to the ROB via the S-link [5].
- the local memory control bus.

In **Table 6** a list of the signals needed are shown. The PU output is considered as a synchronous FIFO, which can be read with a clock of 40MHz in a data driven mode. When the signal for an

event is set from the PU, the controller reads the first word in the FIFO, which indicates the number of words for the event, and sets a countdown counter. The PU design should handle error cases due to wrong assignment of the number of words in the event. All PU designs should comply with these requirements, as well as with the event format as defined in Section 4.2.1.

For the local memory, the MT48LC4M16A2-10 chip from Micron is chosen. It is a Synchronous Dynamic RAM with a configuration of 4 Meg x 16 bit. Only two chips are needed to reach 16MB.

Name	Origin	Function
PU board control bus		
Evnt_ready	PU	when high, indicates that the PU Board has one complete event
Read_ck	ROD	Read synchronous clock input, 40 MHz
Oecmd_b	OC	When low, the 32-bit output is enabled.
		When high, the output bus must be in high impedance state.
Read_en_b	OC	When low indicates to the PU that it must give the next data on the bus, at least 10ns after the low-to-high transition of the Read_ck signal.
Evnt_end	OC	This 25ns signal indicates to the PU that the entire event is read.
The S-link control bus		
Uwen_b	OC	User Write Enable. When low enables data to be transferred to the S-link on the low-to-high transition of the clock.
Uctrl_b	OC	User Control line. When low indicates that the data transmitted is a control word
Udw<1:0>	Not Used	User Data Width line. Define the data width the S-link is to be operated in. This two lines are connected to the ground, which defines 32 bit width
Utest_b	OC	User Test line. When low switches the S-link in test mode
Ureset_b	OC	User Reset line. When low initiates a reset cycle
Ldown_b	S-link	Link Down. When low indicates that the S-link is not operational
Lff_b	S-link	Link Full Flag. After it goes low, up to two more words may be written
Lrl<3:0>	Not used	Driven by the S-link. Link Return lines
The memory control bus		
Cke	OC	Clock enable. When high the clock input on the memory chip is enabled
Cs_b	OC	Chip select. When low the memory's command decoder is enabled
Ras_b, Ras_b, We_b	OC	Command input. These three lines define the command being entered
Dqm<1:0>	Not used	These two lines are connected to the ground, which allows 16-bit bus width on each memory chip
Addr<11:0>	OC	These lines can be the Row address (addr<11:0>) or the Column address (addr<9:0>) depending of the command
Ba<1:0>	OC	These two lines are used to select the bank. There is 4 banks in each chip. Each bank is 1Meg x 16 bit

Table 6 The output controller signals.

For the controller configuration, a 32-bit read-write register is used loaded by the VME interface as shown in **Table 7**.

Bit	Name	Function
0-11	Rowmax<11:0>	Used only when the S-link is disabled and local memory mode enabled, to indicate the maximum memory address for the data. When reached, the current event is readout and then the controller stops.
12-13	Bankmax<1:0>	
14-15	Reserved	
16-23	Mask<1-8>	To mask (disable) single PU boards from the readout.
24	EnLink	Enable the output data stream to the ROB link.
25	EnSpy	Enable the spy mode. Only relevant if the EnLink bit is set, otherwise ignored. In this case a complete event is copied in the local memory to be readout by VME. A refresh mechanism is also set if the event is not read within a time window.
26	Dtm	Enable local data taking mode. Only relevant when the EnLink is not set. When is set the local memory is used to continuously store events. When is not set, a single event is read and then the controller stops.
27	EnTstLink	Enable test link. When set, the Utest_b line in the S-link control bus is activated to put the S-link in the test mode.

28-31	Reserved

Table 7 The output controller configuration register.

The output controller also receives and stores the TTC signals for each event into FIFOs, as shown in **Figure 5**. There are three FIFOs, which are used: one for the BCID, one for the EventID and one for the Trigger Type. In principle for each trigger received a PU event fragment should be found.

2.5.2 The readout sequence

The readout sequence is data driven, and initiates from the L1A information in the TTC FIFO. This way it is guaranteed that for each L1A there will be an event send to the ROB as required. Since the processing time in the events has large fluctuations, it is expected that the PU output buffer is large enough to compensate for this. The OC starts reconstructing an event once all the PU Event Fragments become available.

The readout sequence consists of the following steps:

- Step 0: The sequence starts when all the non masked PU have sent the Evnt_ready signal
- Step 1: The event fragment header is constructed, using the TTC information
- Step 2: The first word in the FIFO of the first PU is read, and a countdown counter is set
- Step 3: The rest of the words for the event are read
- Step 4: The controller checks that the last word is the EventMarker (0xffff).
- Step 5: Steps 2 - 4 are repeated for all the PU
- Step 6: The event fragment trailer is constructed.

The complete output data format is found in Section 4.2.2 .

2.5.3 Output

At the output, the data go either into the ROD-ROB link or into the VME interface.

ROD-ROB link

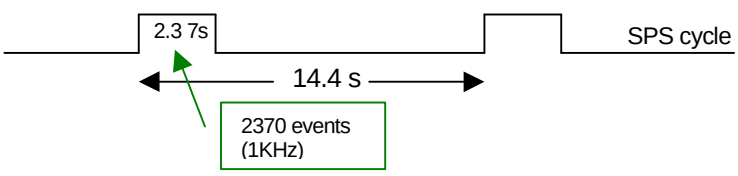
In **Table 8** the available bandwidth for different LVL1 trigger rates is shown as well as the expected typical size of an event.

LVL1 rate	Link speed	Max. # words to transfer	Event size	
			Typical	Margin
75 KHz	40 MHz	533/13.3μs	389(38.8%)	27%
100 KHz	40 MHz	400/10 μs	389(4.1%)	2.7%
	66 MHz	660/10 μs	389(71.8%)	41%

Table 8 The bandwidth margin for the output link for the different LVL1 trigger rates and link speed.

VME interface

In the test beam it is required that the DAQ runs at about 1KHz rate, which implies that about 2000 events will be taken at each SPS burst with a timing as shown in the diagram below:



Using the maximum event size[1] of 1453 words or ~5.67KB, it means that in the output a large buffer of **~13.1MB** should be available. In addition, the VME readout should be fast enough to empty the buffer before the next spill. Assuming that there will be 7 boards to be read out each time[2] this means that the VME and subsequently the DAQ should be able to handle more than **9.1MB/s** sustained rate, which should be feasible with today's computers.

2.6 The VME Interface

In general the ROD module is considered as a slave module. All the actions and commands are controlled by the local CPU. The ROD module can issue interrupts to cause certain actions to happen. In general, the VME interface on the motherboard has to allow the following actions:

Load and configure the motherboard and the PU components

The ROD receives commands and executes certain actions. In this case a general broadcast is needed so that the code or parameters can be loaded simultaneously in all the boards and PU as fast as possible.

Select operating mode

There will be many modes of running: normal data taking, calibration, debugging with data injected by VME. Switching from one mode to the other requires global or partial re-configuration of the ROD and it's PU. It might be that switching from one mode to the other will require a certain number of steps which the VME interface should handle.

Get the calibration or physics data

When the boards are configured for calibration, the data should be send via the VME interface to the host. Also in the test beam the main data flow goes through the VME bus.

Get error messages and flags

The ROD modules will produce various error messages and flags. The VME interface should be able to send these to the host where appropriate actions should take place: board reset, force a busy signal etc. The error messages and flags should be classified in terms of severity and there should be VME Host interrupts on fatal errors.

Get additional monitoring information

The VME interface should allow to read by the VME host the monitoring histograms or counters in the PU.

Mask malfunctioning boards of PU or single ADCs or FEBs

Malfunctioning PU or single ADCs in the data stream should be able to be masked via the VME interface. In this case the masked elements should be properly flagged in the data.

2.6.1 Implementation

There are two components for the VME interface implementation: the main component, which is located in the motherboard and the one in each PU. The block diagram of these components is shown in **Figure 9**.

[1] In the test beam it is probable that we would like to have both the raw data as well as the calculated quantities for all the events, at least during a debugging phase.
[2] Corresponding to 14 FEBs, enough to cover a large area of the detector

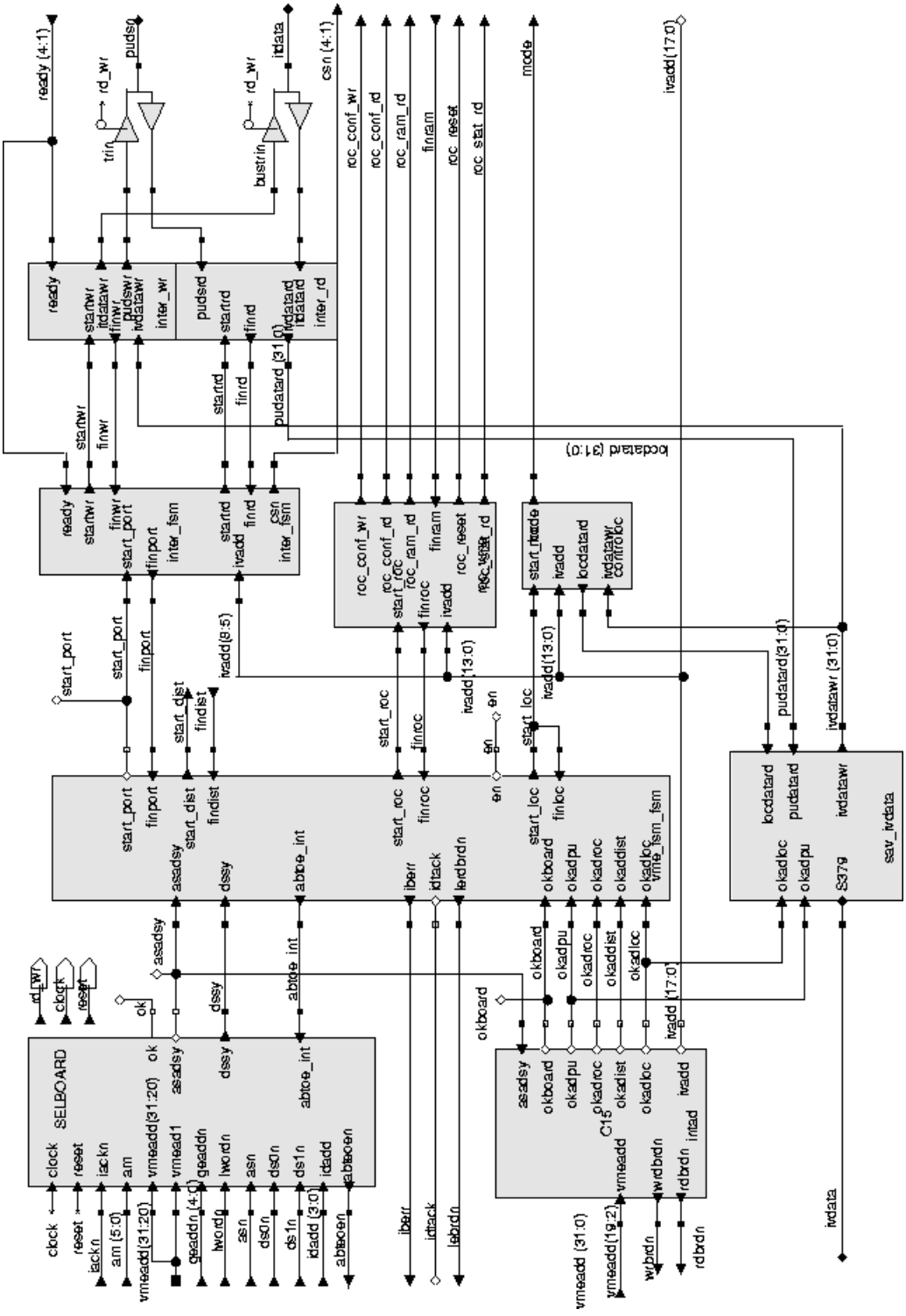


Figure 9 Block diagram for the VME interface implementation.

The communication towards the VME master follows exactly the VME64x standard. For the PU a protocol has been defined as described bellow.

2.6.2 Communication protocol between the motherboard and the PU

For the communication with the PU an 8-bit data bus running at 40MHz is used. The signals between the motherboard and the PU are show in **Table 9**, followed by example read/write cycles.

# Bits	Direction	Polarity	Name	Mode	Function
8	Mb ↔ PU	Normal	Data	Sync	Data multiplexed in 4 bytes. The data are present during 2 clock cycles (50 ns). The PU_Dsn is sent during the 2nd clock when the data are stable. The MSB containing byte is sent first.
1	Mb ↔ PU	Inversed	Data Strobe	Sync	Is a clock enable and must have a front edge of of the clock (TTC_Clk) during its validity.
1	Mb → PU	Inversed	Chip Select	Sync	Is sent by the motherboard to initiate a read or write cycle
1	Mb ← PU	Normal	Ready	Sync	Is sent by the PU to indicate that the reception register is free. At the beginning of the cycle it says that a transaction is possible (from MB to PU), while at the end of a transaction it indicates that a new word can be send. In this way a FIFO is not necessary between the communication port and the DSP. The motherboard waits until the PU drives the ready
1	Mb → PU	Normal	Read/Write		Image of VME write signal
5	Mb → PU	Normal	Internal VME Address		Image of VME Add(bits 2:6)
2	Mb ← PU	Hi	Interrupt		To be sent to VME Interrupt
1	Mb ← PU	Hi	Busy		To be sent to the BUSY OR
1	Mb → PU	Hi	Reset		General reset of PU
5	Mb ↔ PU	Hi	JTAG		Programming pins of FPGA or DSP
26	**Total**				

Table 9 Communication signals between the PU and the Mother board VME modules.

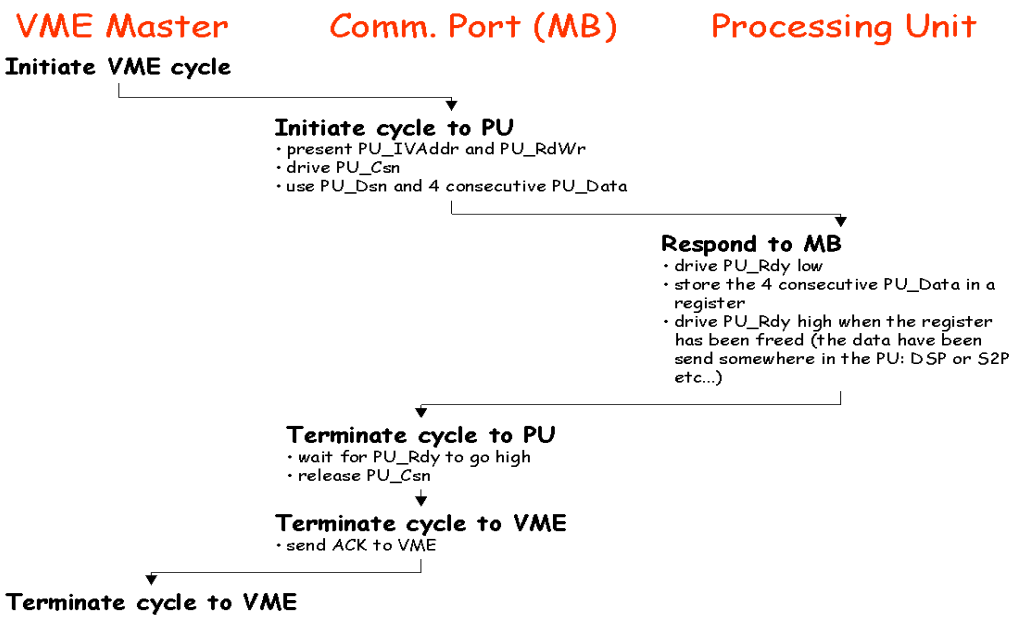


Figure 10. Typical VME write cycle to the PU.

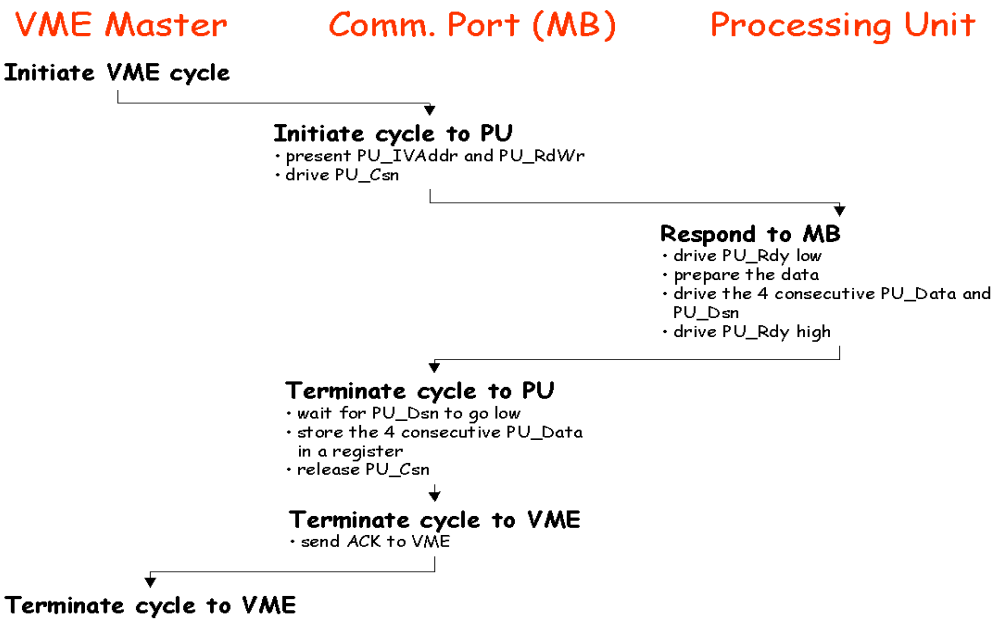


Figure 11 Typical VME read cycle from the PU.

2.6.3 VME address mode

The VME address (A24) has 32 bits (VMEADD 0-31) and the general addressing schema is shown in **Table 10**. Bits <23:31> are used for the geographical address and ROD identification. For all the components of the board bits <2:6> are reserved for the internal function implementation. The exact implementation depens on the particular PU design.

VMEADD	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
General	Geogr. address					ROD ID					Component															Address space						
Proc. Unit																1						PU ID				Address space						
																						4	3	2	1							
Data Distr.															1											Address space						
Output Controller														1			RAM	RAM Memory address								Address space						
TTC Ctrlr.													1													Address space						
Misc,												1														Address space						
Local											1															Address space						

Table 10 The VME addressing schema for the various components.

Each PU is identified using a PU id field (bits <7:10>). A broadcast mode is also possible when the PU_id field is set to 0. As an example such a mode will be particular useful when loading the code for the DSPs or other FPGAs.

2.7 The Power Distribution

2.7.1 Motherboard

For the components in the motherboard +3.3V power foreseen in VME64x standard will be used. In the cases where additional power is needed it will be made with local regulators starting from the +3.3V.

2.7.2 Processing Units

For the PU, a dedicated power will be provided.

It is estimated that each PU will need about 2-3A @ 3.3V. A DC-DC converter in the motherboard will provide either +3.3V or +5V starting from the +48V available as auxiliary power in the VME64x standard.

A front panel switch is foreseen to cut the power to all the PU, thus allowing fast exchange of the daughter cards without having to shutdown the whole crate.

3 The DSP Processing Unit

In this document no exact description of the PU is given since its design depends on the choice of the DSP processor. However, some basic functions and requirements are described which all designs should follow.

3.1 Requirements for the PU Design

[1] Buffer the data in both the input and the output. To absorb the fluctuations due to different event sizes and algorithms used, it is estimated that a buffer of about 100 events of the average size (see **Table 13**) would be needed in both cases.

[2] Evaluate the energy, time and quality of fit for all the channels. It is possible that for the channels with low energy the time or the quality of fit are calculated using simple algorithm while for the channels above a threshold more complicated algorithms might be used.

[3] Comply with the data format as defined in Section 4.2. All the word counting should be used in 32-bit words.

[4] Monitor the data and the hardware using simple histograms as defined in 4.2.1.

[5] Do the first step in the analysis of the calibration data.

[6] The PU design should comply with all the requirements defined for the input/output and VME interfaces with the motherboard. In particular it should be able to receive and store the TTC information without any possible loss.

[7] Should be able to produce the BUSY signal when the buffers are almost full (using a programmable value, which needs to be tuned for the BUSY latency), or by software as required for the calibration.

[8] Handle parity or synchronisation errors in the data and issue the appropriate signals.

As shown in **Figure 1** the basic design, for the PU contains three major components:

- the **serial to parallel converter (S2P)** which reads and checks the input FEB data and sends them to the DSP,
- the **DSP chip with it's local memory** (internal and/or external), connected to the input/output FIFOs,
- the **VME interface** through which all the control of the board is done. It is also used to send monitoring or debugging information or histograms to the local CPU (via the VME interface on the motherboard).

The exact implementation of these depends on the choice of the DSP and the way the data are treated. The RODDemo motherboard design has clear and rather general interfaces with the PU thus allowing maximum flexibility for the developers. In the following paragraphs some basic functions for the PU are presented in order to have some degree of uniformity.

3.2 Error handling

There are basically two types of errors in the ROD board:

- Errors in the data flow: either in the data format or synchronisation errors
- Hardware or software errors of the board: corrupted memory or program, etc…

The PU should be able to handle both types of errors. As a general rule, for any error an interrupt followed with a status word describing the error should be generated. In addition, dataflow errors should be properly flagged in the data using specific bits as foreseen in the data format. The local ROD CPU will take the interrupt and trigger the corresponding actions.

3.2.1 Data flow errors

Parity Check

At the input stage the parity of each data word should be checked. If an error is found, then bit 15 (MSB) of the word should be set to 1 (normally set to 0 from the FEB). A total count of the parity errors of the event should be made and if non-zero, a flag should be set.

Given that the parity errors in the experiment should be rather small as a first step it's suggested that no analysis is made for those events, but an empty output event is send to the output. As a second step the parity errors in the control words and in the data can be separated thus recovering some of the events.

A monitoring of the parity errors should be made (see section bellow).

Synchronisation

The data flow in the PU follows two general guidelines/constraints:

- **The TTC information received is the reference.** For each TTC trigger information that the PU receives it has to output an event fragment either complete or just the header in the case of errors.
- **The TTC information for an event arrives before the FEB data:** It is expected that the path TTC → FEB → ROD will take 1ms more than the TTC → ROD[3].

These two guidelines are followed in the board design.

Each PU receives data from one FEB link as well as the TTC information. There are several possible error conditions that can cause synchronisation loss in this case:

- the FEB board has missed a trigger, therefore no data will be available,
- the ROD has received a fake trigger,
- due to parity errors in the links, the beginning of the event word is missed or the BCID word in either the FEB data or the TTC data is corrupted.

As general rule the PU board will take the TTC information that it receives as reference and will try to match it with the incoming FEB data by comparing the BCID numbers. In the case where the

[3] This is most probably correct, but depends on the actual location of the ROD crates. In any case it can be made as such by adjusting the cables or delays.

comparison fails, the data should be properly flagged, an interrupt in the local CPU should be made to indicate the type of error, but the dataflow should continue. It's up to the design of the PU if such data will be analysed but in any case an event should be produced in the output.

Since there is not a simple way to recover from synchronisation errors, it is foreseen that a periodic (or on request) reset will be available in ATLAS to avoid loosing too many data. This reset signal will clear all the buffers in the FEB system as well as the TTC FIFO's in the PU. The PU design should be able to receive this and perform the appropriate actions.

3.2.2 Hardware errors

If hardware or software errors in the PU are identified, an interrupt should be generated followed with the appropriate status to the local CPU. The exact type of errors depends on the design of the PU.

3.3 Histograms – Data monitor

To monitor the data flow in the PU some histograms should be booked and filled. A proposed list is shown bellow:

100% of the events

- Parity errors per event: 3 bins (1,2, >2) $\times$ 32-bits = 12 Bytes

  (BER of 10^{-12} in the link corresponds to 1 error per 156Mevents, or ~0.5 hour)
- Gain of each channel: 2 bins (med, low) $\times$ 16-bits $\times$ 64 channels = 256 Bytes

  (with 10% of the channels in medium gain, the histogram will be filled every ~6 s)

~1% of the events

- Baseline monitoring (high gain): 128 bins $\times$ 16-bits $\times$ 64 channels = 16 KB

  (at least 65 s to be filled in one word)

Channel activity histograms

Presumably there will be two thresholds for the energy and time calculation. One where the energy and coarse time are calculated and one where precise time and quality of fit are calculated. In each case the following histograms should be filled:

Above first threshold (10% of the channels)

- Proportion: 1 bin $\times$ 16-bits $\times$ 64 channels = 128 Bytes
- Baseline monitoring: 128 bins $\times$ 16-bits $\times$ 64 channels = 16 KB
- Amplitude, time calculation: 2 $\times$ 128 bins $\times$ 16-bits $\times$ 64 channels = 32 KB

Above second threshold (1% of the channels)

- Proportion: 1 bin $\times$ 16-bits $\times$ 64 channels = 128 Bytes
- Baseline monitoring: 128 bins $\times$ 16-bits $\times$ 64 channels = 16 KB
- Amplitude, time, quality of fit: 3 $\times$ 128 bins $\times$ 16-bits $\times$ 64 channels = 48 KB

In the worse case the histograms can be read every ~65s before being in overflow.

In summary, for this basic set a total memory of ~129 KB is required for the histograms which can be emptied with a frequency of 0.2 Hz. Assuming an VME bus bandwidth of 10 MB, the read cycle will require ~13ms.

3.4 PU design

There are currently undergoing two separate designs for the PU, one based on the Analog Devices SHARC DSP and one with the Texas Instruments C6x DSP family. Details of their designs will be available in separate documents and in the Largon web pages.

4 The Data Format

4.1 Input Data format

The data from the FEB link are organised per ADC (there are 16 ADCs per FEB board) in 16-bit words[6]. For the default case where 5 samples are read per channel, 50 words are sent per event per ADC.

For more details please refer to [6].

4.2 Output Data Format

For the output there are two formats that need to be defined:

- The Intermediate Data Format at the output of each PU.
- The ROD Data Format for the data send to the ROB.

4.2.1 Intermediate data format – PU event fragment

Each PU formats the data into four 32-bit word blocks as sown bellow, framed with one header word which contains the total size of the event[4] and an end of event marker word which has a fixed value 0x0EOE.

Total number of words in the event (=sum of all blocks)	
Block #0	Control and status word (fixed length)
Block #1	Energy sums (fixed length)
Block #2	Energy per channel (fixed length)
Block #3	Time and quality of fit information (variable length)
Block #4	Raw data (variable length)
Block #5	RADD words (variable length)
End of event marker (0x00000EOE)	

Table 11 Intermediate (PU→MB) ROD event data format.

In the following a detailed description of each data block is given.

In case of errors it is sufficient that the PU sends only the first block with the appropriate flags.

Fixed block #0

It contains the header and control word information and will be present in all the events.

	31	15
1	No words in block #0	No words in block #1
2	No words in block #2	No words in block #3
3	No words in block #4	No words in block #5

[4] Only the data words should be counted, without including the trailer 0xEOE word.

4	Mask	FEB Number	TTC_BCID	
5	CTL3 (ADC 2)		CTL3 (ADC1)	
6	CTL3 (ADC 4)		CTL3 (ADC3)	
7	CTL3 (ADC 6)		CTL3 (ADC 5)	
8	CTL3 (ADC 8)		CTL3 (ADC 7)	
9	Control Bits (ADC 4)	Control Bits (ADC 3)	Control Bits (ADC 2)	Control Bits (ADC 1)
10	Control Bits (ADC 8)	Control Bits (ADC 7)	Control Bits (ADC 6)	Control Bits (ADC 5)
11	Status Flags		Number of channels	
12	Extra info in block #3			
13	Extra info in block #3			
13	Extra info in block #4			
15	Extra info in block #4			

Note:

- The *Mask* bit field indicates which of the ADCs for the PU are masked from the readout.
- The Control Bits words correspond to bits 7-11 in the CTRL1 word at the input data.
- The meaning of the *StatusFlags* bit-field needs to be defined.
- Setting a bit to 1 in the words 12-15, indicates that additional information for that channel in the specified block is provided.

Fixed block #1

It contains the energy sum information, present for all the events

	31
1	Energy sum 1
2	Energy sum 2
3	Energy sum 3
4	Energy sum 4
5	Energy sum 5

Fixed block #2

It contains the energy information, and will be present for all the events.

	31
1	Energy Channel 1
2	Energy Channel 2
…	……
64	Energy Channel 64

Note:

- The energy for each channel will be a 32-bit IEEE floating point word.

Variable block #3

It contains the time and quality of fit information, and will be present only in the case there are channels with energy above threshold.

1	Time Channel 1	Sflag 1	Fit Channel 1
2	Time Channel 2	Sflag 2	Fit Channel 2
…	……..		
63	Time Channel 63	Sflag 63	Fit Channel 63
64	Time Channel 64	Sflag 64	Fit Channel 64

Note:

- The **maximum** total length of this block is 64.
- The time here is a 16 bit signed integer with LSB corresponding to 5ps. This gives a range of ±160ns, which is much more than needed.
- The fit will be a 9-bit integer (always positive).
- The flag bits are described below:

Bit 0	the energy is over threshold for this channel
Bit 1	the fit is out of range
Bit 2	the baseline is out of range
Bit 3-4	the gain of the channel
Bit 5	reserved

Variable block #4

	31	15
1	Channel 1 Sample 2	Channel 1 Sample 1
2	Channel 1 Sample 4	Channel 1 Sample 3
3	Xxxxxxxxxxxxxxxx	Channel 1 Sample 5
4	Channel 2 Sample 2	Channel 2 Sample 1
5	Channel 2 Sample 4	Channel 2 Sample 3
6	Xxxxxxxxxxxxxxxx	Channel 2 Sample 5
……	……	
190	Channel 64 Sample 2	Channel 64 Sample 1
191	Channel 64 Sample 4	Channel 64 Sample 3
192	Xxxxxxxxxxxxxxxx	Channel 64 Sample 5

Note:

- The **maximum** length of this block is 192 words in the case of 5 samples.

Fixed block #5

	31	16
1	RADD (ADC 1. Sampl 2)	RADD(ADC 1 Sampl 1)
2	RADD (ADC 1. Sampl 4)	RADD(ADC 1 Sampl 3)
3	RADD (ADC 2. Sampl 1)	RADD(ADC 1 Sampl 5)
4	RADD (ADC 2. Sampl 3)	RADD(ADC 2 Sampl 2)
5	RADD (ADC 2. Sampl 5)	RADD(ADC 2 Sampl 4)
6	RADD (ADC 3. Sampl 2)	RADD(ADC 3 Sampl 1)
7	RADD (ADC 3. Sampl 4)	RADD(ADC 3 Sampl 3)
8	RADD (ADC 4. Sampl 1)	RADD(ADC 3 Sampl 5)
9	RADD (ADC 4. Sampl 3)	RADD(ADC 4 Sampl 2)
10	RADD (ADC 4. Sampl 5)	RADD(ADC 4 Sampl 4)
11	RADD (ADC 5. Sampl 2)	RADD(ADC 5 Sampl 1)
12	RADD (ADC 5. Sampl 4)	RADD(ADC 5 Sampl 3)
13	RADD (ADC 6. Sampl 1)	RADD(ADC 5 Sampl 5)
14	RADD (ADC 6. Sampl 3)	RADD(ADC 6 Sampl 2)

15	RADD (ADC 6. Sampl 5)	RADD(ADC 6 Sampl 4)
16	RADD (ADC 7. Sampl 2)	RADD(ADC 7 Sampl 1)
17	RADD (ADC 7. Sampl 4)	RADD(ADC 7 Sampl 3)
18	RADD (ADC 8. Sampl 1)	RADD(ADC 7 Sampl 5)
19	RADD (ADC 8. Sampl 3)	RADD(ADC 8 Sampl 2)
20	RADD (ADC 8. Sampl 5)	RADD(ADC 8 Sampl 4)

In the table below the size of the different data blocks in the PU event fragment.

Block	Detector Block	Length	Maximum Size	Typical Size
-	Header	-	1	1
#0	Control Words	Fixed	15	15
#1	Energy sums	Fixed	5	5
#2	Energy	Fixed	64	64
#3	Time, Status & Quality of Fit	Variable	64	7
#4	Raw Data Information	Variable	192	3
#5	Read Addresses	Variable	20	0
-	End of event marker	-	1	1
		Total	**362(360)**	**96(94)**

Table 12 The maximum and typical event size of the Intermediate Data Format for one PU in units of 32-bit words.

Note:

The typical size is calculated assuming that ~10% of the channels have energy above threshold for which the precise time needs to be calculated, ~1% of the channels have high energy for which the raw data need to be available, and no RADD words are send. The header and end of event marker words won't be transmitted in the output

4.2.2 ROD event fragment format

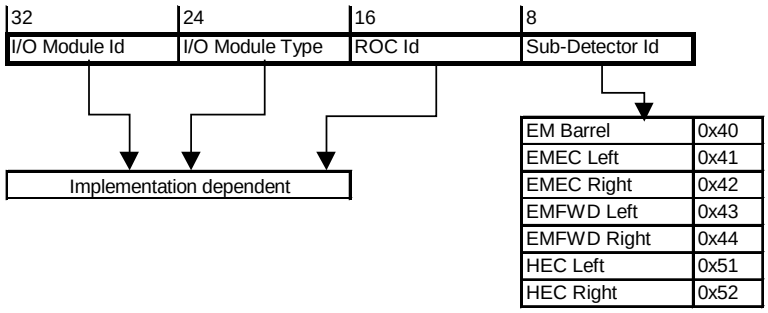
The output controller has to construct complete event fragments, according to the ATLAS DAQ/EF Prototype –1, event format [7]:

		32 31 0			
		0	**Beginning of fragment (0x00000B0F)**		
	Header	1	Start of header marker (0xEEEEEEEE)		
		1	Header size (0x08)		
		1	Format version number		
		1	Source Identifier		
		1	Level 1 ID		
		1	Bunch crossing ID		
		1	Level 1 Trigger Type		
		1	Detector Event Type		
D a	Status word	1	No of PU	PU Mask	Detector Format Version Number
	Processing Unit 1	1	Fixed Block #0		
		1	Fixed Block #1		
		1	Fixed Block #2		
		1	Variable Block #3		
		1	Variable Block #4		
		1	Variable Block #5		
	Processing Unit 2	1	Fixed Block #0		
		1	Fixed Block #1		
		1	Fixed Block #2		
		1	Variable Block #3		

		1	Variable Block #4
		1	Variable Block #5
	Processing Unit 3	1	Fixed Block #0
		1	Fixed Block #1
		1	Fixed Block #2
		1	Variable Block #3
		1	Variable Block #4
		1	Variable Block #5
	Processing Unit 4	1	Fixed Block #0
		1	Fixed Block #1
		1	Fixed Block #2
		1	Variable Block #3
		1	Variable Block #4
		1	Variable Block #5
	Status Word	1	Status flag from Output Controller
Trailer		1	Number of Status Elements
		1	Number of Data Elements
		1	Status Block Position
		0	**End of Fragment (0x0E0F)**

Note:

- The *Format Version number* is a 32-bit integer number loaded by VME at configuration time. It defines the version of the ROD data fragment header NOT the format version of the detector data.
- The *Source Identifier* is the word that defines the fragment. It is sub-divided as sown below:



The exact implementation of these words will depend on the detector readout organisation.

- The *Level 1 ID*, *Bunch Crossing ID*, *Level 1 Trigger Type* are delivered by the TTC system.
- The *Detector Event Type* this is a sub-detector specific event type flag. It will be send by VME at configuration time.
- NoPU indicates the number of PU present in the data.
- PU Mask is the 8-bit pattern with defines which of the PU of the ROD board is read out. The Pus are numbered from 1 to N with #1 the one on the top of the ROD board.
- Detector Format Version Number indicates the format version for the detector data block.
- *Number of status elements*, which gives the total length of the detector status block.
- *Number of data elements*, which gives the sum of the detector blocks #1 to #5.
- *Status block position*, which is set to 0, to indicate that the status block precedes the data, as it is in this case.

- The *Status flag from Output Controller* is used to mark events where the data from the PU are not read correctly. For example when there is a mismatch between the number of words to read from a PU and the end of event marker.

Event Fragment Block	Length	Maximum	Typical
Fragment Header	-	8	8
Status word	-	1	1
Detector Block #0 (control)	Fixed	60 (4× 15)	60 (4× 15)
Detector Block #1	Fixed	20 (4× 5)	20 (4× 5)
Detector Block #2	Variable	256 (4× 64)	256 (4× 64)
Detector Block #3	Variable	256 (4× 64)	28 (4 × 7)
Detector Block #4	Variable	768 (4× 192)	12 (4× 3)
Detector Block #5	Variable	80 (4× 20)	0 (4× 0)
Status word	-	1	1
Trailer	-	3	3
	Total	**1453**	**389**

Table 13 The ROD Event Fragment size in units of 32-bit words.

4.3 Calibration Data Format

In the case of calibration events the ROD board will do the first step in the data analysis and then send the results in the local CPU for further treatment. In this case, each PU will have to send the mean, rms and number of events read for each channel for each step in the calibration process[5].

In this case the proposed format per PU is shown in **Table 14**.

	32		15	
0	No of events	No Step	Status Flags	PU ID
1	Sum E Channel 1 Sample 1		Sum E^2 Channel 1 Sample 1	
2	Sum E Channel 1 Sample 2		Sum E^2 Channel 1 Sample 2	
3	Sum E Channel 1 Sample 3		Sum E^2 Channel 1 Sample 3	
4	Sum E Channel 1 Sample 4		Sum E^2 Channel 1 Sample 4	
5	Sum E Channel 1 Sample 5		Sum E^2 Channel 1 Sample 5	
6	Sum E Channel 2 Sample 1		Sum E^2 Channel 2 Sample 1	
7	Sum E Channel 2 Sample 2		Sum E^2 Channel 2 Sample 2	
8	Sum E Channel 2 Sample 3		Sum E^2 Channel 2 Sample 3	
9	Sum E Channel 2 Sample 4		Sum E^2 Channel 2 Sample 4	
10	Sum E Channel 2 Sample 5		Sum E^2 Channel 2 Sample 5	
	.....			
316	Sum E Channel 64 Sample 1		Sum E^2 Channel 64 Sample 1	
317	Sum E Channel 64 Sample 2		Sum E^2 Channel 64 Sample 2	
318	Sum E Channel 64 Sample 3		Sum E^2 Channel 64 Sample 3	
319	Sum E Channel 64 Sample 4		Sum E^2 Channel 64 Sample 4	
320	Sum E Channel 64 Sample 5		Sum E^2 Channel 64 Sample 5	

Table 14 The PU event fragment for the calibration events.

In this case the Output controller will read the data from each PU and send them via the VME interface into the local CPU.

The event fragment size per PU in the case of 5 samples will be 321 32-bit words corresponding to 1.25 KB. Assuming a VME bus speed of 10 MB/s this will require 0.1ms, which is sufficient.

[5] A "*step*" is defined with a set of parameters, which are loaded in the calibration board. The ROD will know at boot time how many events are expected per step and should calculate the mean and rms of the channels for this number of events.

5 Pinout Allocation

5.1 Motherboard Connectors

5.1.1 VME interface (connector P1)

raw	z	a	b	c	d
1	MPR(*)	D00	BBSY*(*)	D08	VPC(1) (*)
2	GND	D01	BCLR*(*)	D09	GND(1)
3	MCLK(*)	D02	ACFAIL*(*)	D10	+V1
4	GND	D03	BG0IN*(*)	D11	+V2
5	MSD(*)	D04	BG0OUT*(*)	D12	RsvU(*)
6	GND	D05	BG1IN*(*)	D13	-V1
7	MMD(*)	D06	BG1OUT*(*)	D14	-V2
8	GND	D07	BG2IN*(*)	D15	RsvU(*)
9	MCTL(*)	GND	BG2OUT*(*)	GND	GAP*(*)
10	GND	SYSCLK(*)	BG3IN*(*)	SYSFAIL*(*)	GA0*
11	RESP*(*)	GND	BG3OUT*(*)	BERR*	GA1*
12	GND	DS1*	BR0*(*)	SYSRESET*	+3,3V
13	RsvBus(*)	DS0*	BR1*(*)	LWORD*	GA2*
14	GND	WRITE*	BR2*(*)	AM5	+3,3V
15	RsvBus(*)	GND	BR3*(*)	A23	GA3*
16	GND	DTACK*	AM0	A22	+3,3V
17	RsvBus(*)	GND	AM1	A21	GA4*
18	GND	AS*	AM2	A20	+3,3V
19	RsvBus(*)	GND	AM3	A19	RsvBus(*)
20	GND	IACK*	GND	A18	+3,3V
21	RsvBus(*)	IACKIN*	SERA(*)	A17	RsvBus(*)
22	GND	IACKOUT*	SERB(*)	A16	+3,3V
23	RsvBus(*)	AM4	GND	A15	RsvBus(*)
24	GND	A07	IRQ7*(*)	A14	+3,3v
25	RsvBus(*)	A06	IRQ6*	A13	RsvBus(*)
26	GND	A05	IRQ5*	A12	+3,3V
27	RsvBus(*)	A04	IRQ4*	A11	LI/I*(*)
28	GND	A03	IRQ3*	A10	+3,3V
29	RsvBus(*)	A02	IRQ2*	A09	LI/O*(*)
30	GND	A01	IRQ1*	A08	+3,3V
31	RsvBus(*)	-12V(*)	+5VSTDBY(*)	+12V(*)	GND(1)
32	GND	+5V	+5V	+5V	VPC(1) (*)

The pins marked with (1) are MFBL (mate-first-break-last) pins. Pins marked with (*) are not connected in the motherboard.

5.1.2 Input links (connector P2)

row	z	a	b	c	d
1	FEB1_D3	FEB1_D2	+5V(*)	FEB1_D1	FEB1_D0
2	GND	FEB1_D5	GND	GND	FEB1_D4
3	FEB1_D7	GND	RETRY*	FEB1_D6	GND
4	GND	FEB1_D10	A24	FEB1_D9	FEB1_D8
5	FEB1_D14	FEB1_D13	A25	FEB1_D12	FEB1_D11
6	GND	FEB1_D17	A26	FEB1_D16	FEB1_D15
7	FEB1_D20	FEB1_D19	A27	GND	FEB1_D18

8	GND	FEB1_D23	A28	FEB1_D22	FEB1_D21
9	FEB1_D25	GND	A29	FEB1_D24	GND
10	GND	FEB1_D28	A30	FEB1_D27	FEB1_D26
11	FLAG1(6)	FEB1_D31	A31	FEB1_D30	FEB1_D29
12	GND	FLAG1_INT	GND	GND	FLAG1(4)
13	STRB1_40M	GND	+5V(*)	FLAG1(0)	FLAG1(5)
14	GND	GOOD1_DATA	D16	FLAG1(1)	(#)RST1_RXBR
15	(#)ATLAS_CLK	RST1_TXGLA	D17	FLAG1(2)	GND
16	GND	RST1_TXGLB	D18	FLAG1(3)	GND
17	FEB2_D3	FEB2_D2	D19	FEB2_D1	FEB2_D0
18	GND	FEB2_D5	D20	GND	FEB2_D4
19	FEB2_D7	GND	D21	FEB2_D6	GND
20	GND	FEB2_D10	D22	FEB2_D9	FEB2_D8
21	FEB2_D14	FEB2_D13	D23	FEB2_D12	FEB2_D11
22	GND	FEB2_D17	GND	FEB2_D16	FEB2_D15
23	FEB2_D20	FEB2_D19	D24	GND	FEB2_D18
24	GND	FEB2_D23	D25	FEB2_D22	FEB2_D21
25	FEB2_D25	GND	D26	FEB2_D24	GND
26	GND	FEB2_D28	D27	FEB2_D27	FEB2_D26
27	FLAG2(6)	FEB2_D31	D28	FEB2_D30	FEB2_D29
28	GND	FLAG2_INT	D29	GND	FLAG2(4)
29	STRB2_40M	GND	D30	FLAG2(0)	FLAG2(5)
30	GND	GOOD2_DATA	D31	FLAG2(1)	(#)RST1_RXBR
31	(#)ATLAS_CLK	RST2_TXGLA	GND	FLAG2(2)	GND(1)
32	GND	RST2_TXGLB	+5V(*)	FLAG2(3)	VPC(1) (*)

Note that data pins in row b are not available on the transition module, but the +5V and GND pins are.

The pins marked with (#) are output towards the transition module.

The pins marked with (*) are not connected in the motherboard.

5.1.3 Output Links (connector P3)

Row	z	a	b	c	d
1	(2) (*)	(2) (*)	+5V(*)	(2) (*)	(2) (*)
2	GND	(2) (*)	GND	GND	(2) (*)
3	(2) (*)	GND	TTC_Plus	(2) (*)	GND
4	GND	(2) (*)	GND	(2) (*)	(2) (*)
5	(2) (*)	(2) (*)	GND	(2) (*)	(2) (*)
6	GND	(2) (*)	TTC_Minus	(2) (*)	(2) (*)
7	(2) (*)	(2) (*)	GND	GND	(2) (*)
8	GND	(2) (*)	GND	(2) (*)	(2) (*)
9	(2) (*)	GND	BUSY0	(2) (*)	GND
10	GND	(2) (*)	BUSY1	(2) (*)	(2) (*)
11	(2) (*)	(2) (*)	BUSY2	(2) (*)	(2) (*)
12	GND	(2) (*)	GND	GND	(2) (*)
13	(2) (*)	GND	+5V(*)	(2) (*)	(2) (*)
14	GND	(2) (*)	BUSY3	(2) (*)	(2) (*)
15	(2) (*)	(2) (*)	BUSY4	(2) (*)	GND
16	GND	(2) (*)	BUSY5	(2) (*)	GND
17	UD3	UD2	BUSY6	UD1	UD0
18	GND	UD5	BUSY7	GND	UD4
19	UD7	GND	BUSY8	UD6	GND
20	GND	UD10	BUSY9	UD9	UD8

21	UD14	UD13	BUSY10	UD12	UD11
22	GND	UD17	GND	UD16	UD15
23	UD20	UD19	BUSY11	GND	UD18
24	GND	UD23	BUSY12	UD22	UD21
25	UD25	GND	BUSY13	UD24	GND
26	GND	UD28	BUSY14	UD27	UD26
27	UCTRL#	UD31	BUSY15	UD30	UD29
28	GND	UWEN#	BUSY16	GND	UDW0
29	UCLK	GND	BUSY17	UDW1	UTEST#
30	GND	LDOWN#	BUSY18	LFF#	URESET#
31	LRL1	NC (3) (*)	GND	LRL0	GND
32	GND	LRL3	+5V(*)	LRL2	VPC(*)

Note: This assumes that we use a P2 type backplane will be used the P3 position. In this case, raw b will be used to send the TTC signals and the BUSY. For each ROD module the BUSY will be assigned a particular pin according to its geographical address.

(2) Reserved for an additional input or output link. Are not connected normally.

(3) NC: do not connect.

Pins marked with (*) are not connected in the motherboard.

5.2 Mezzanine Board Connectors

The list of signals between the motherboard and the PU is show in **Table 15**.

Line	Name		No of pins
Input data			
Data	FEB_Data		16
Clock	FEB_Clk		1
Link Error status	FEB_LnkSt		1
Link reset	FEB_LnkRst		1
		Total	**19**
TTC Information			
Clock	TTC_Clock		1
Data (BCID, Trigger Type)	TTC_Data		8
Write for BCID	TTC_BCIDWr		1
Write for Trigger Type	TTC_TtypeWr		1
Reset	TTC_Rstn		1
		Total	**12**
PU Control			
Data	PU_Data		8
Register address	PU_IVAddr		5
Read/Write	PU_RdWr		1
Data strobe	PU_Dsn		1
PU ready	PU_Rdy		1
PU reset	PU_Rstn		1
PU chip select	PU_Csn		1
		Total	**18**
Miscellaneous			
BUSY	BUSY		1
Interrupt request 1	PU_Irq1		1
Interrupt request 2	PU_Irq2		1
		Total	**3**
Output			
Data	FIFO_Data		32

Read enable	FIFO_Rdenn		1
Clock	FIFO_RdClk		1
Output enable	FIFO_Oen		1
Event ready	FIFO_EvtRdy		1
Event end	OC_EvtEnd		1
		Total	**37**
JTAG			
Test data input	TDI		1
Test data output	TDO		1
Test clock	TCK		1
Test mode select	TMS		1
Test reset	TRST		1
		Total	**5**
Reserved for test purposes			16
	Total number of signal pins		**110**
Power			
Ground	GND		40
+3.3 V (regulated from 48V)	+3.3V		18
		Total	**58**
		Total number of pins	**168**

Table 15 List of signals between the motherboard and the PU.

The signals are transferred in the PU through two 84-pin connectors. The pinout allocation is shown bellow. Pin #1 corresponds to the top left pin of the connector in the motherboard.

Connector A (left side)

1	+3.3V	GND	2
	GND	TTC_Clk	
	TTC_BCIDWr	GND	
	TTC_TTypeWr	TTC_D0	
	GND	TTC_D1	10
11	TTC_D2	TTC_D3	12
	TTC_D4	GND	
	TTC_D6	TTC_D5	
	+3.3V	TTC_D7	
	TTC_Rstn	GND	20
21	TDI(*)	PU_Irq1	22
	GND	PU_Irq2	
	TCK(*)	+3.3V	
	TDO(*)	BUSY	
	TRST(*)	TMS(*)	30
31	GND	GND	
	TST_D1	TST_D0	
	TST_D3	TST_D2	
	TST_D5	TST_D4	
	TST_D7	TST_D6	40
41	TST_D9	TST_D8	
	TST_D11	TST_D10	
	TST_D13	TST_D12	
	TST_D15	TST_D14	
	+3.3V	+3.3V	50
51	+3.3V	+3.3V	
	GND	FEB_D0	
	FEB_D1	FEB_D2	
	GND	FEB_D3	

	FEB_D4	GND	60
61	GND	FEB_D5	
	FEB_D6	FEB_D7	
	GND	FEB_D8	
	FEB_D9	GND	
	GND	FEB_D10	70
71	FEB_D11	FEB_D12	
	GND	FEB_D13	
	FEB_D14	+3.3V	
	+3.3V	FEB_D15	
	FEB_LnkSt	GND	80
81	GND	FEB_Clk	
	FEB_LnkRst	GND	84

(*) For the first version of the demonstrator, the JTAG pins are not connected.

Connector B (right side)

1	+3.3V	PU_Dsn	2
	PU_RdWr	+3.3V	
	GND	PU_D0	
	PU_D1	PU_D2	
	PU_D3	PU_D4	10
11	PU_D5	GND	12
	GND	PU_D6	
	PU_Rdy	PU_D7	
	PU_Rstn	PU_Csn	
	PU_IvAddr0	GND	20
21	PU_IvAddr1	PU_IvAddr2	22
	PU_IvAddr3	PU_IvAddr4	
	+3.3V	OC_EvtEnd	
	FIFO_RdClk	GND	
	+3.3V	FIFO_EvtRdy	30
31	FIFO_Rdenn	FIFO_D31	
	GND	FIFO_D30	
	FIFO_D29	+3.3V	
	GND	FIFO_D28	
	FIFO_D26	FIFO_D27	40
41	GND	FIFO_D25	
	FIFO_D24	GND	
	GND	FIFO_D23	
	FIFO_D21	FIFO_D22	
	GND	FIFO_D20	50
51	FIFO_D19	+3.3V	
	+3.3V	FIFO_D18	
	FIFO_D16	FIFO_D17	
	GND	FIFO_D15	
	FIFO_D14	GND	60
61	GND	FIFO_D13	
	FIFO_D11	FIFO_D12	
	GND	FIFO_D10	
	FIFO_D9	GND	
	GND	FIFO_D8	70
71	FIFO_D6	FIFO_D7	
	GND	FIFO_D5	
	FIFO_D4	GND	

	GND	FIFO_D3	
	FIFO_D1	FIFO_D2	80
81	+3.3V	FIFO_D0	
	FIFO_Oen	+3.3V	84

5.3 TTCrx mezzanine connectors

The list of signals between the motherboard and the TTCrx mezzanine board are listed bellow. The connectors are as seen from the top of the motherboard.

Connector A

2	EVTCNTLSTR	EVCNTHSTR	1
	GND	BCNTSTR	
	BCNT<1>	BCNT<0>	
	BCNT<3>	BCNT<2>	
10	GND	GND	9
	BCNT<5>	BCNT<4>	
	BCNT<7>	BCNT<6>	
	BCNT<9>	BCNT<8>	
	BCNT<11>	BCNT<10>	
20	GND	GND	19
	DOUT<1>	DOUTSTR	
	DOUT<3>	DOUT<0>	
	DOUT<5>	DOUT<2>	
	DOUT<7>	DOUT<4>	
30	SDA	DOUT<6>	29
	SCL	VDD	
	VDD	GND	
	GND	CLOCK40DES1	
	VDD	GND	
40	VDD	BCNTRES	39
	L1ACCEPT	EVCNTRES	
	GND	GND	
	SUBADDR<0>	SUBADDR<2>	
	SUBADDR<1>	SUBADDR<4>	
50	SUBADDR<3>	SUBADDR<6>	49
	SUBADDR<5>	GND	
	GND	BRCSTSTR1	
	SUBADDR<7>	BRCST<2>	
	BRCST<3>	BRCST<4>	
60	BRCST<5>	BRCST<6>	59
	BRCST<7>	GND	
	GND	TTCREADY	
	RESET_B(*)	DBERRSTR	
	SINERRSTR	JTAGTDI	
70	JTAGTDO	JTAGMS	69
72	JTAGCK	JTAGRST_B(*)	71

(*)RESET_B and JTAGRST_B are active low signals.

Connector B

2	GND	GND	1
	GND	TTC_PLUS	
	GND	GND	

	GND	TTC_MINUS	
10	GND	GND	9

6 VME interface

Addressing schema for the motherboard components

The VME addresses for the motherboard components are shown bellow. The R/W direction is with respect to the crate CPU: R=the CPU reads, W=the CPU writes the command/data.

Address	R/W	Function	VME Data	
Component: Local (VMEADD: bit <21>=1, bit<6:2>=Address)				
0	R	Status	<3:0>	PU ready signal
1	W	Reset		
3	W/R	Configure	<0>	1=VME simulates TTC and FEB data, 0=
			<1>	TTC Information from: 1=front panel, 0=TTCrx
Component: MISC (VMEADD: bit <20>=1, bit<6:2>=Address)				
0	R	Status		
1	W/R	Enable	<0>	BUSY: 1=enabled, 0=dissabled
			<1>	Interrupts: 1=enabled, 0=dissabled
2	R	FEB Link status		
XXXX	W/R	Interrupt module x ID	<31:0>	ID
XXXX	W/R	Mask interrupt x	<11>	status: 0=masked, 1=active
			<10:8>	value of bits <1:7> in binary
			<7>	1=connected to VME interrupt 7, 0=masked
			<6>	1=connected to VME interrupt 6, 0=masked
			<5>	1=connected to VME interrupt 5, 0=masked
			<4>	1=connected to VME interrupt 4, 0=masked
			<3>	1=connected to VME interrupt 3, 0=masked
			<2>	1=connected to VME interrupt 2, 0=masked
			<1>	1=connected to VME interrupt 1, 0=masked
Component: TTC Controller (VMEADD: bit <19>=1, bit<6:2>=Address)				
0	R	Status		
1	W	Clear Status		
2	W	Event ID Write	<23:0>	value
3	W	BCID Write	<11:0>	value
4	W	Ttype Write	<7:0>	value
Component: Output Controller (VMEADD: bit <18>=1, bit<6:2>=Address)				
0	R	Status		
1	W	Reset		
2	W/R	Configure		
4	W	Format Version No	<15:0>	value
8	W	Source identifier		
10	W	Detector event type		
12	W	Detector format version No		
Component: RAM (VMEADD: bit <18:15>=1001, bit<14:7>=RAM address, bit<6:2>=Address)				
0	R	Read RAM	<31:0>	RAM memory value
Component: Data Distributer (VMEADD: bit <17>=1, bit<6:2>=Address)				
0	R	VME data to PU	<31:0>	data value: <31:16> to PU #2,4 and <15:0> to PU 1,3

7 Upgrade Path

There are several points that need to be further looked at before the final ROD boards are made, but it will depend on the performance of the first prototypes.

Acronyms

BCID	Bunch Counter Identification
DSP	Digital Signal Processor
FEB	Front-End Board
OC	Output Controller
PU	DSP Processing Unit
ROB	Readout Buffer
ROD	Readout Driver
TTC	Trigger Timing Control
TBM	Trigger BUSY Module

References

[1] The ATLAS Liquid Argon Calorimeter Technical Design Report
(http://www.cern.ch/Atlas/GROUPS/LIQARGON/TDR/)

[2] A 9U Transition Module for the ROD Demonstrator
(http://www.particle.kth.se/~stefan/ROD_TM)

[3] The TTCrx Reference Manual
(http://www.cern.ch/Atlas/GROUPS/FRONTEND/Ttc1.htm)

[4] The Output Controller for the ROD Demonstrator board.
(http://atlasinfo.cern.ch/Atlas/GROUPS/LIQARGON/ROD/docs/OutputController_v1.0.doc)

[5] The S-link interface for the ROD-ROB links in ATLAS.
(http://www.cern.ch/HSI/s-link/)

[6] Format for the Data read out from the front-end boards, Note ATL-LAL-ES-1.0.
(http://www.phys.ualberta.ca/~electronics/DataFormat/index.html)

[7] The event format in the ATLAS DAQ/EF Prototype -1, ATL-DAQ-98-129