ATLAS Internal Note
TILECAL-NO-033
16 November 1994

TILECAL Data Aquisition System for the ATLAS Test Beam Experiments

Renius Arsenescu
CERN - PPE Division, CH 1211 Genève 23 Switzerland

November 16, 1994

Contents

1 Introduction

RD-34 uses for the 1994 test beam a more performant Local Aquisition System in order to satisfy the requirements of the ATLAS tests . This system is designed and organized differently than the one used in 1993.

Compared with the old DAQ System, the new one accepts higher trigger rates, has a better data format, it is more flexible and provides a full compatibility with the ATLAS General Aquisition (performed by RD-13). The monitoring tasks are also more flexible and complete.

2 Short description of ATLAS Test Beam Experiment Data Aquisition System

ATLAS H8 Test Beam DAQ is a complexe and flexible system designed upon the multiple criteria required by the experiment. It integrates all local detector DAQ systems, it is easy configurable in order to include or exclude a given detector and can cope with different types of runs: physiscs events calibration or combined.

As it was allready suggested , the system uses a General Aquisition which reads and process data from the different subdetectors Local Aquisitions. To achieve this, a common interface was designed.

The block configuration of the full DAQ system is presented in fig. 1.

For each subdetector was defined a standard configuration implemented as a VME structure. This includes:

- - an individual Memory Area divided into a number of fixed size Logical Buffers capable to hold event data from a particular subdetector
- - a CPU unit
- - an Interrupt Manager for SPS and Trigger signals
- - a Readout Module
- - a Vertical Bus Interface

Almost the same configuration (excepting the R/O module) is used by the Event Builder which reads data from subdetectors via the Vertical Bus.

Each local DAQ system is able to act upon the SPS signals (" Start of Burst" and "End of Burst") and the Trigger signals, generating a Busy signal on the occurence of each of these ones. The Busy signal is kept active for the whole duration of the specific procedure in progress. The Busy associated with a given signal is interpreted by the Event Builder as a signal of the same nature,

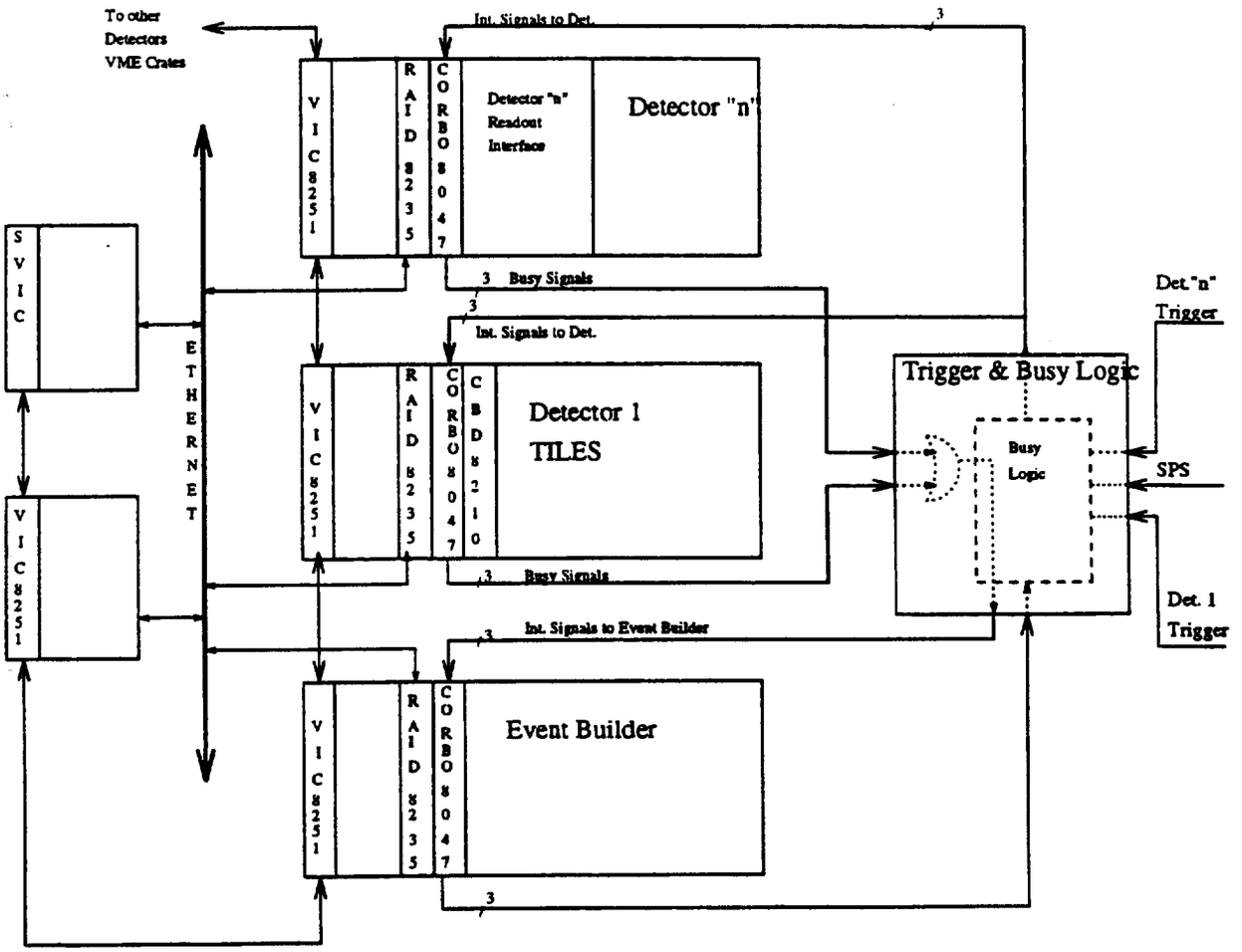

Figure 1: The Configuration of the Experiment DAQ

on the occurance of which an Event Builder Busy signal is generated. This last one should inhibit all the activities of the Local Aquisitions. A new readout operation can start only when the E.B. Busy is released. The individual buffers of the subdetectors are emptied at the end of each burst, when data are transfered by DMA to the General Aquisition which performs the event building. The individual memory buffers are organised in such a way that data from the same physical event provided by each detector are correlated.

The monitoring tasks can be plugged in at local level or at the Event Builder level. In particular for TileCal it is the second case.

The General Aquisition provides also the data recording system. Data is stored on magnetic support (Exabyte Cartridges).

3 Requirements for the TILECAL Data Aquisition System

3.1 The timing of the readout by respect to the burst

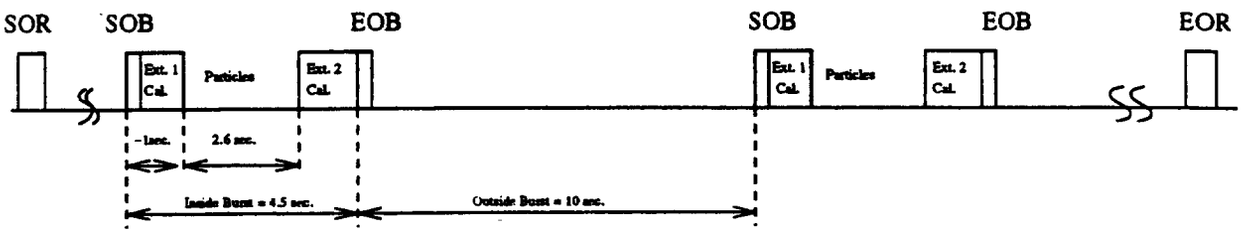

Figure 2: The Structure of the Burst

The timing diagram of the readout is presented in fig. 2

The SPS "Spill" duration (4.5 seconds) is marked by two signals: "Start of Burst" and "End of Burst". The bunch of particles arrives some one second after the SoB signal and finishes some 1 second before the EoB signal (its duration is about 2.6 seconds). This particular structure of the burst is used in order to perform the readout for physics events and for calibration procedures. The readout process is started by the occurence of a "Start of Run" signal at a moment "to". No readout is performed till the SoB signal appears. The first second after SoB and the last one before EoB (which are called "extensions of the burst")are used for special readout procedures as Pedestals, or Laser Calibrations. The real burst (when particles arrive) is used for reading out the physics events and, optionally, Pedestals and Laser Calibration events. After EoB the system remains inactive till the next SoB.

For the particular case of PPG Calibration procedure the whole Spill duration is used.

3.2 Requirements imposed by the calorimeter

During the 1994 Test Beam, RD-34 will test five modules of the Hadron Calorimeter. Each module being read by 40 Photomultiplier Tubes, this extends the number of signal lines to 200. It is required that the signals on each line to be read both nonamplified and amplified (k = 7.5) in order to cover the wide dynamic range of the detector response for different kind of particles in the beam. This allows an analog-to-digital conversion on only 12 bits with a sensitivity of 4 ADC counts per picocoulomb of collected charge.

Data are recorded without pedestal or zero substraction.

The DAQ is able to analyse the calorimeter respose for Physics events, for Laser and Electronic (PPG) Calibration Procedures and for Pedestals. Excepting

the PPG Calibration, all these procedures can be performed individually , during dedicated runs, or can be combined. The DAQ System provides also information about the beam parameters corelated with the Physics events.

The monitoring task is required for the online check of the calorimeter response, the beam quality, for calculating pedestal values and calibration constants.

3.3 Requirements imposed by the General Aquisition

From the General Aquisition point of view the local DAQ System is supposed to cope with the standards imposed for signals, for the memory structure and for the data format.

As it was allready mentioned, the local aquisition generates interrupts and Busy signals in response to the standard SPS signals (SOB and EOB) and to a Trigger signal which is common for all detectors. The system treats electrical pulse signals and generates electrical level (Busy) signals. The specific Trigger signals for the Hadron Calorimeter (from physics events and/or calibration procedures) are "summed" (by the mean of an OR function) with Trigger signals from other detectors in a Global Trigger Busy Logic and become part of the common Trigger signals.

The Hardware configuration of the Local Aquisition Sytem is totally independent from the Event Builder and other detectors. It includes an independent 32 MBytes memory area which resides on the CPU card and an interface to the common Vertical Bus.

The memory management is done on a Shared Memory Principle. A memory segment is created and mapped and, afterwords, both the Local and the General Aquisitions are accessing it (the local system for filling the buffer and the Event Builder for dumping it).

Data are organised in a standard way as a tree structure. There is a Fixed Header and a User Header for the Full Event and a table of pointers which give the adresses for all the detectors data areas. Each area has again a Fixed Event Header, an User Header and a block of pointers to the blocks of data associated with the subdetectors which form a detector.

4 The TILECAL DAQ System

Seen from the experiment level, TILECAL (RD-34) Local DAQ System is one part of the tree like structure. Though, it is functionally independent and provides a hardware and a software interface to the full system.

4.1 The Hardware organisation

TILECAL Local Aquisition is implemented on 3 level:
- The CPU and the main process controllers are at the VME level.
- The Readout configuration is implemented at CAMAC level.
- The Local Trigger and Busy Logic is implemented at NIM level.

4.1.1 VME Level

The RD-34 VME structure includes a Processor Unit with independent memory area, an Interrupt Manager, a CAMAC Branch Controller, and a Vertical Bus Interface (fig. 3).

The Processor Unit is a CES RAID 8235 module , a risc VME processor running real-time UNIX (TC/IX operating system). For the particular application we discuss now only the VME accessing ability of the module is exploited. The physical mapping of the bus is programable by software. The VME (and VSB) interfaces can be accessed by both the R3000 CPU and the DMA controller by the me an of two distinct address spaces. These spaces are divided in spaces of 16MB . The local aquisition runs on the RAID processor. The other VME modules are controlled by this one via the VME bus.

The CES VIC 8251 module is a Vertical Bus Interface which allows a transparent connection of up to 15 VME crates on the same (VMV) bus. It includes master/slave VMV and VME interfaces (the VSB slave interface is not used for this application). It provides a transfer speed up to 8Mb/s. The internal buffer memory capacity is 512 Kb. It also supports muti-processing on the VMV bus.

RCB 8047 CORBO is a CES Interrupt Manager module. It can handle up to four independent interrupt signals (on four independent channels). It generates a VME interrupt on the occurance of each signal , is able to count the events and it controls the dead time. On the occurance of a signal on one of the four channels a Busy signal is asserted together with the VME interrupt. The Busy signal can be cleared only by a VME access. No other trigger signal is accepted wile the Busy signal is asserted. The module's function mode can be programmed by configuring the Control and Status Register.

The CAMAC branch driver is a CES CBD 8210 module. Up to seven CAMAC crates can be driven via this module from the VME processor. It allows 16-bit and 24-bit transfers and can handle graded LAM requests in two modes.

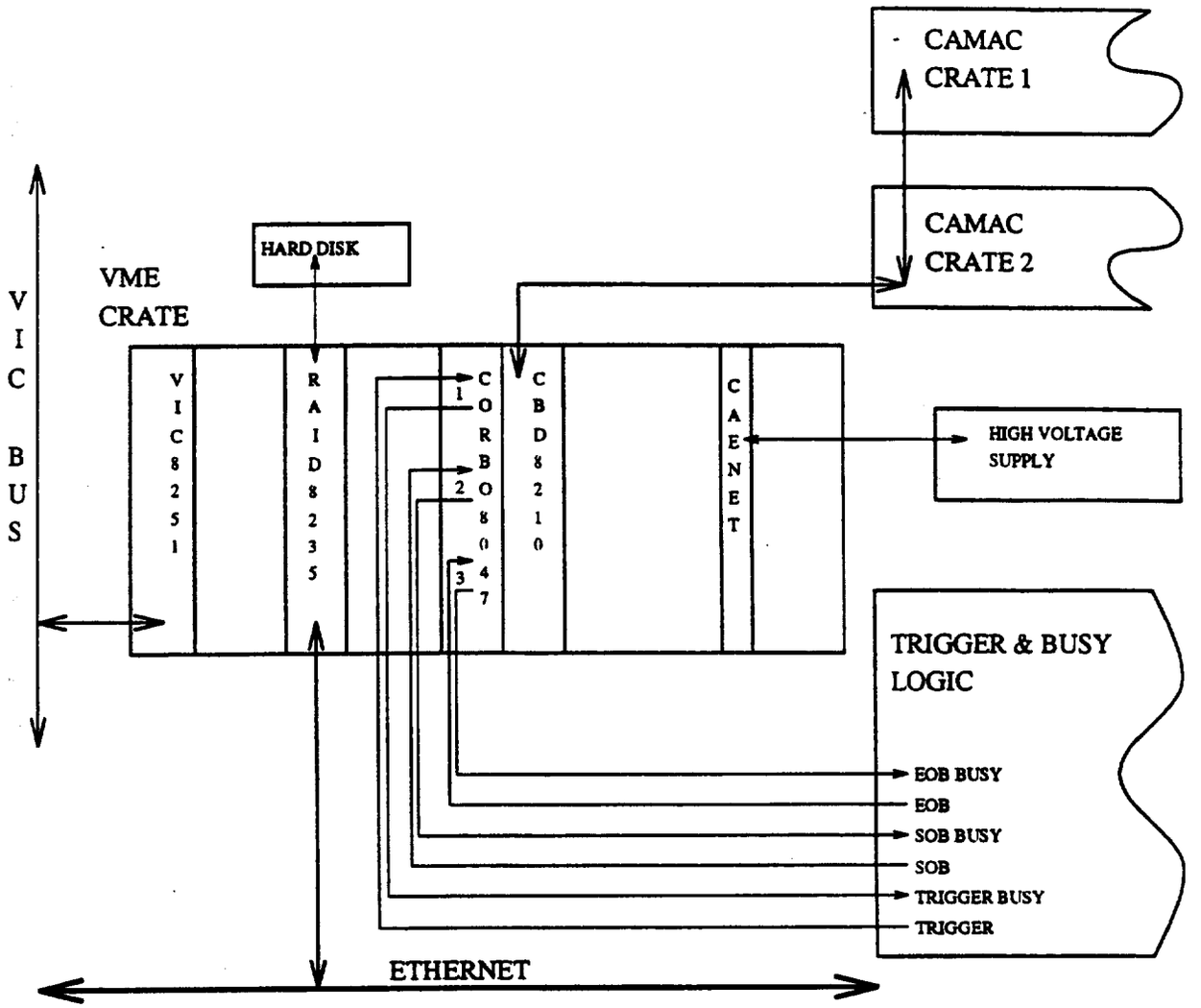

Figure 3: The VME Configuration

For the local aquisition task the interrupt facilities on the GL signals are not used (for reasons of speed).

4.1.2 CAMAC Level

The basic level readout for the TileCal detector is implemented as a CAMAC structure (fig. 4).

There are two independent CAMAC sections:

- one for the calorimeter readout
- one for the auxiliary readout(Beam-Elements) and control (Trigger Mask, PPG Calibraion control etc.)

The calorimeter readout is implemented using seven CAEN 205 ADC modules. These are 32 channels charge ADC-s providing a 12-bit convertion range on two paths, one with x1-gain and the other with x7.5-gain. This provides the advantage of a 15-bit like conversion on a simpler configuration. The input charge is first converted to voltage when coincident with a Gate signal and then this voltage is converted to number in parallel by two different gain paths. The digial convertion is done by succesive aproximations. The two 12 bits resulting words are stored in a RAM memory. When the conversion finishes for all 32 channels a CAMAC LAM signal will be set to "1" signaling that data can be read. The 2*32 words can be read via the CAMAC bus in sequential order. A 65-th CAMAC reading operation clears the module which will now be ready for a new conversion.

The gate signal width can be set between 100ns an 5μs.

The full scale input charges are 112.5pc(for the 15 bit dynamic range) and 900pC (for the 12 bit dynamic range).

The conversion gain is $\simeq$30cnts/pC and $\simeq$4cnts/pC respectively.

The conversion time for the 32 channels is about 1.6ms.

With the seven modules we cover 224 signal paths, from which we use 200 for the calorimeter read-out.

Three other paths are used for the LASER monitoring diodes.

The other CAMAC section contains the Beam Elements read-out and some auxiliary control modules.

For the Beam Counters a LeCroy 2249 12 channels ADC is used. Only three channels are used for the 3 counters. The charge-to-digital conversion is done on 10 bits with 256 pC full scale input charge and 0.25pC/cnts sensitivity. The conversion time is 60μs (the 12 channels are operational in paralel). The gate width allowed is between 10ns to 2μs.

For the Beam Chambers there are used two LeCroy 4208 8 channels TDC modules. These can provide a 23 bits(plus 1 bit for sign) word for each channel, so covering a time domain value of $\pm$8.3ms with a resolution of $\pm$1ns. For each module the conversion is done in paralel for the 8 channels if the signals are coincident with a unique time window. The two modules used in our configuration are daisy-chained in order to have the same window.

- The Beam Elements ADC and TDC-s are used optionally when TILECAL is in stand alone configuration in the beam.

In order to identify the nature of the trigger signal treated at a different moments, we use a Pattern Unit (SEN 2026 - Strobed Input Register) which encodes on a 12 bit word the truth value of the 12 input channels.

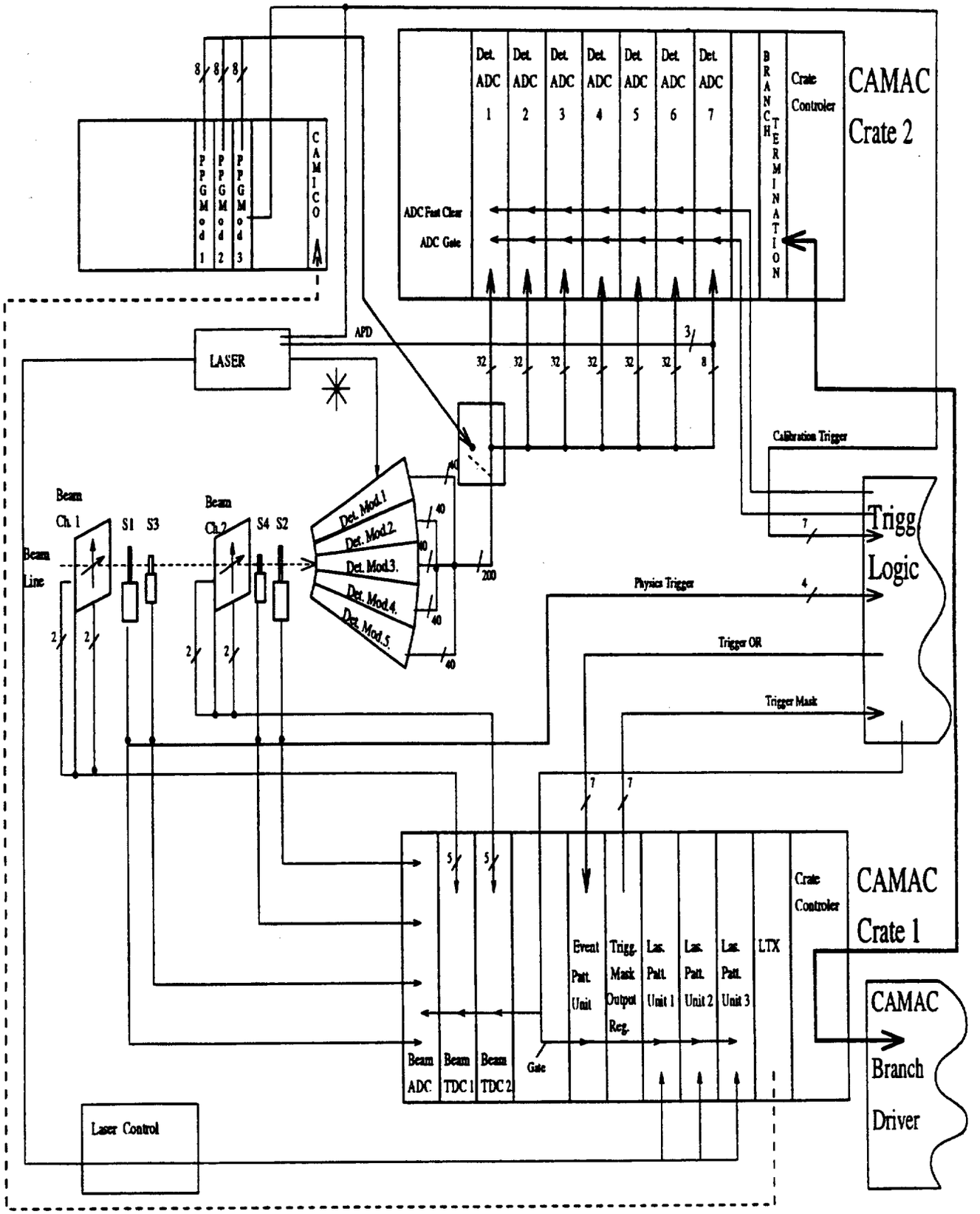

Figure 4: The CAMAC ReadOut and Control Configuration

A CAMAC 12 bits Output Register is used to select specific triggers for differents types of runs. According to the word wrote via CAMAC into the register, there can be found different truth values on the 12 outputs which can be used to veto the trigger signals. The bit significance is identical with the one of the Pattern Unit.

For the special PPG Calibration procedures it is needed to access by remote control another CAMAC crate which lays near the detector. For this we use a CAMAC Transmitter (a SEN LTX 111 module).

The readout is performed on the occurance of each trigger signal accepted by the Busy Logic. The reading out moment is detected by testing the LAM signal on all the detector ADC-s(which are the slowest); at this moment all the CAMAC elements (including the Beam Elements and the auxiliary ones) are read and cleared sequentially.

4.1.3 NIM Level

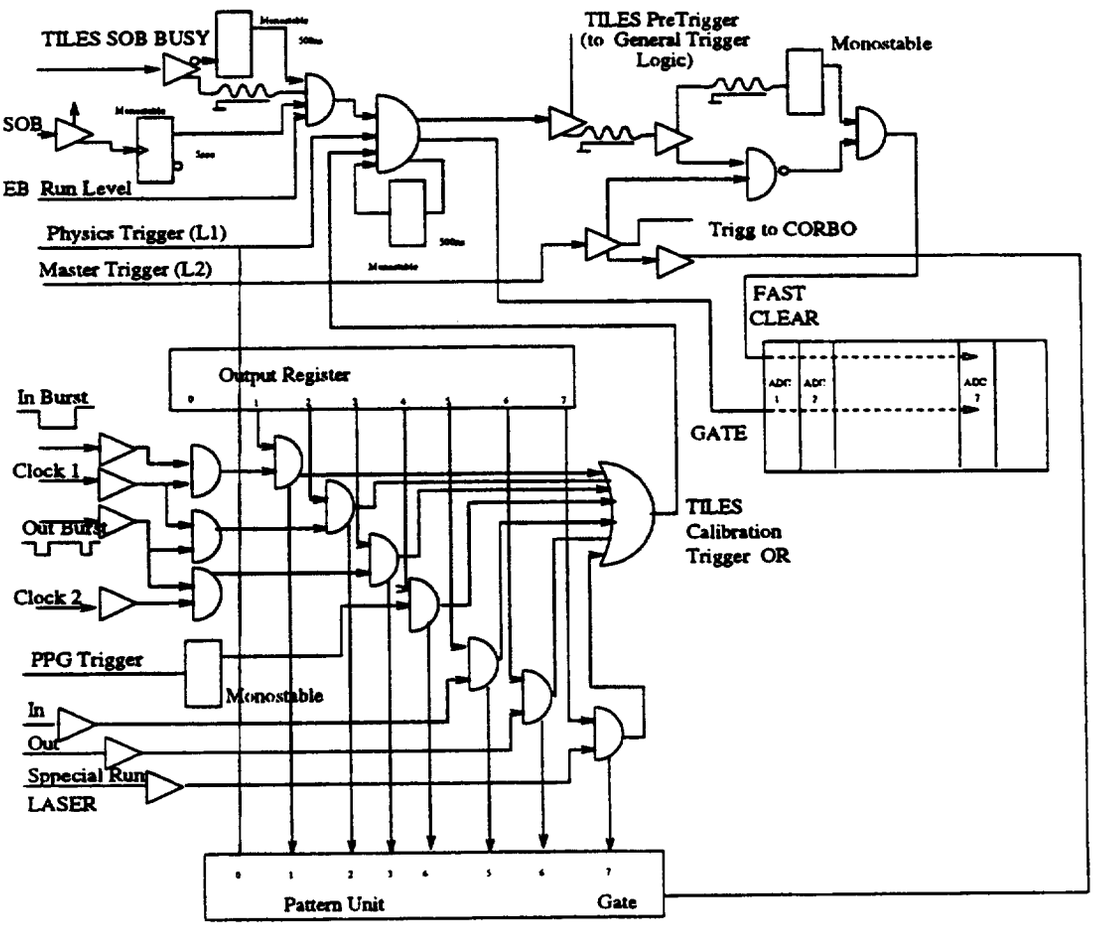

Figure 5: Local Trigger and Busy Logic

The Local Trigger and Busy Logic is implemented at NIM level. The idea is to have a single trigger path for all TileCal specific triggers and to provide a full compatibility with the General Trigger and Busy Logic. The calibration triggers used by the detector are vetoed by the mean of "AND" gates (coincidence units) with the trigger mask given by the CAMAC Output Register. Then, the different "conditioned" triggers are put together (also with the "physics events" trigger, which is not conditioned) into an "OR" logical function (fig. 5).

When the TILECAL detector is alone in the beam line, the "physics Events" trigger is obtained from the coincidence of the three Beam Counters, conditioned by the presence of the Burst "window" and the Ready signals from the Event Builder and the Local Aquisition.

The unique trigger signal which results gives the Gate signal for the local read-out. The same signal is sent to the General Trigger and Busy Logic as "Tiles Pre-trigger". There it is "OR-ed" with other detectors "Pre-triggers" and it is further conditioned by the standard SPS signals and the Ready signals. The signal comes back as the Master Trigger signal which is the same for all detectors. (For the General Trigger and Busy logic please reffer to RD13 Internal Note No.125.)

If a local trigger is not accepted by the General Trigger Logic, a Fast Clear signal will be generated by the Local Logic, preventing in this way the TILECAL detector ADC-s to treat a false signal.

The nature of the trigger treated at different moments is detected by the Pattern Unit as the coincidence of the specific trigger with the Master Trigger.

4.1.4 The PPG Hardware

The PPG Hardware configuration is shown in fig. 6.

The PPG Calibration procedure consists in injecting a fixed electrical charge on each of the detector ADC-s inputs and measuring the response. After this, the charge value is changed and the measurement is repeated. The charge values are in a logarithmic succession. This provides an absolute calibration of the ADC channels and also their intercalibration. Practically, this is done by decoupling the Photomultiplier Tubes from the ADC-s inputs and connecting the charge injectors. For this we provide a DC voltage (about 100V), called the PPG HV, to the CAMICO module in the PPG CAMAC crate. This module, on a specific comand sent by remote, will divide this voltage with different factors and will then send it to the 3 PPG modules. Using another remote command, the PPG modules will produce a short pulse and send it on one PPG channel at the time. Each PPG channel is connected to a PPG box which contains the switches for the signal path and a resistive divider. The calibration pulse will be so divided by 10 and sent on ten ADC inputs at the time. When producing a calibration pulse, the PPG modules will generate also a trigger signal which is sent to the Local Trigger Logic as "PPG Trigger". All the signal paths are 50

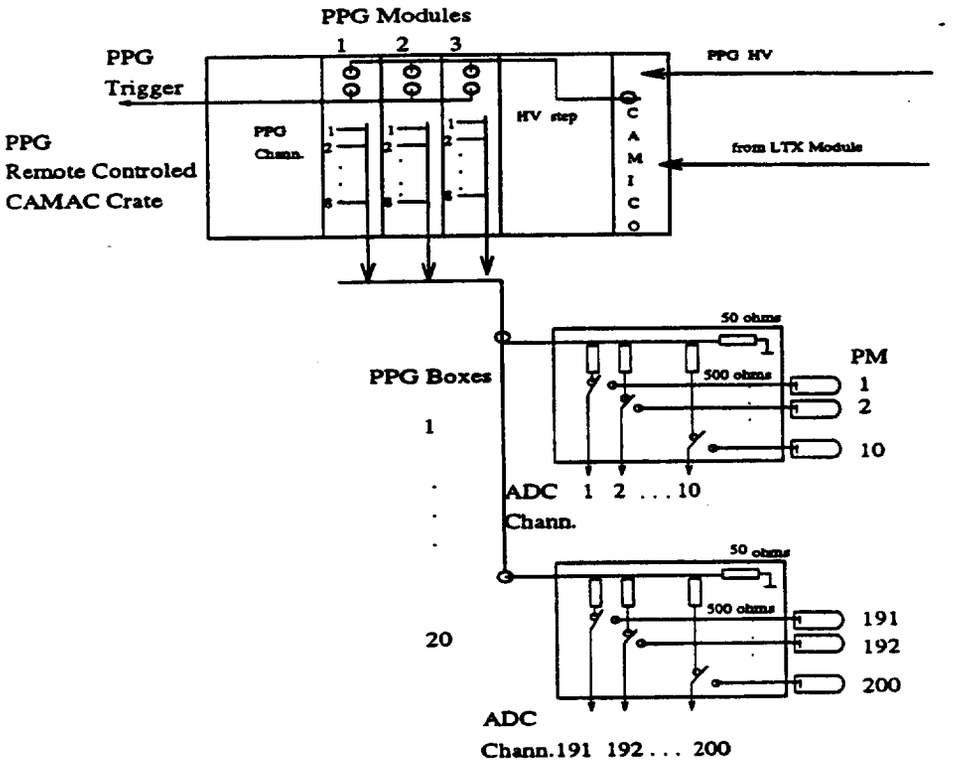

Figure 6: The PPG Hardware

ohms impedance and need to be adapted.

4.2 The Finite State Machine

The behaviour of the Local DAQ System is controlled by a Finite State Machine (fig. 7).

The Finite State Machine is defined by a list of states (represented as rectangles in the FSM diagram) and list of "signals" that can be accepted. On the occurance of a "signal" the system executes a transition between two consecutive states (the transitions are represented as arrows). The "signals" can be interrupt requests attached upon electrical (hardware) signals or simple software commands (software signals). One can associate to each transition a specific routine (which we call "action") so that that, finally, the system will navigate between its states according to the signals that it receives, executing particular procedures if called by these ones.

4.2.1 Normal data taking

- The fundamental state is S0 (IDLE) in which the system waits for the beginning of a new run.

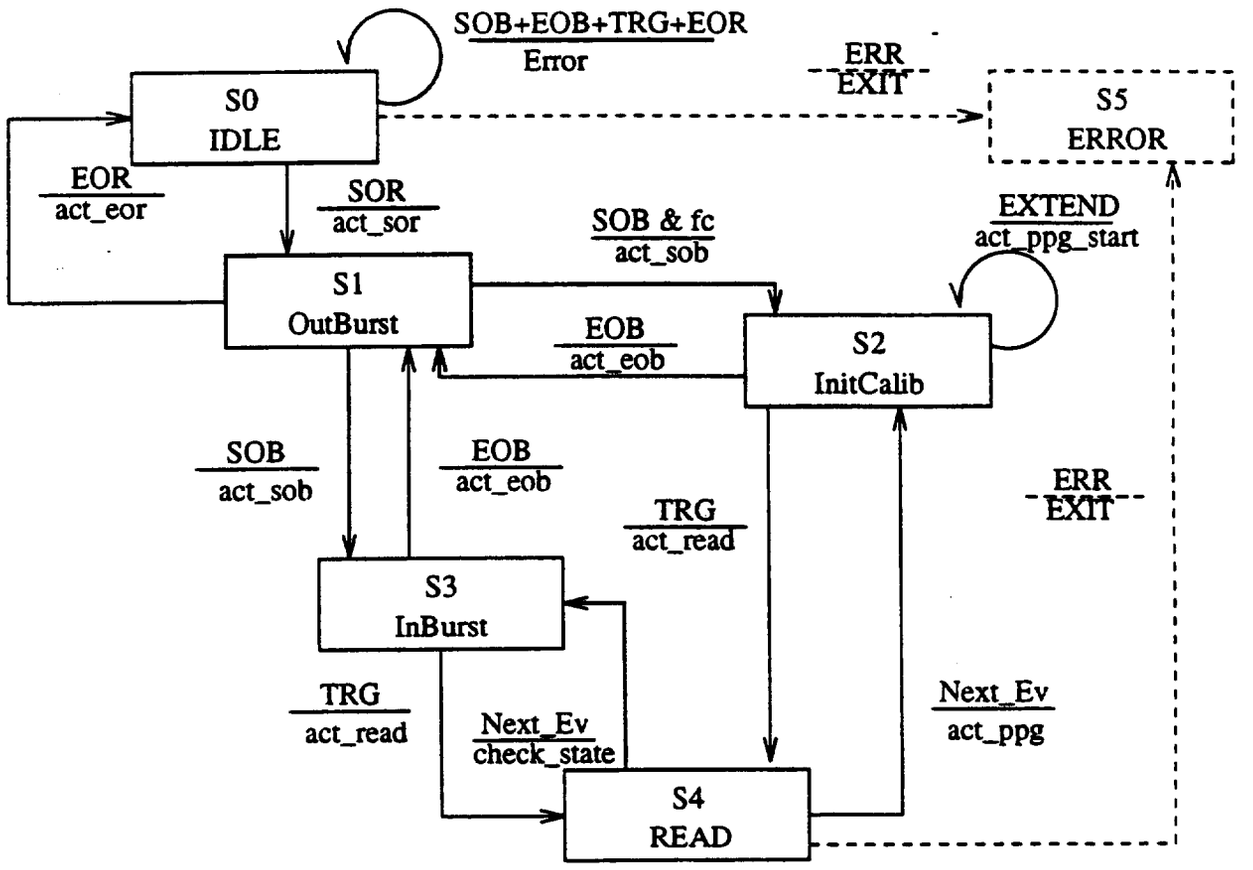

Figure 7: Finite State Machine

- When the run starts, the system receives a "Start of Run" soft signal and makes a transition from S0 to S1 (OUTBURST) executing the action "act_sor". No other signal is accepted. The system is now ready for the SPS signals.
- If a "Start of Burst" hardware signal arrives from SPS, the system will jump to state S3(INBURST) executing the action "act_sob". The system is ready to treat "Trigger" signals.
- If an "End of Run" software signal arrives within S3 the system will go back in state S0 executing the action "act_eor". No other signals are accepted.
- If a "Trigger" hardware signal occurs within state S3, there will be a transition to the state S4 (READ) and the action "act_read" will be executed. By "act_read" the system is reading data from the calorimeter and the beam elements.
- When this procedure is finished a soft signal "Next-Event" is generated which causes the transition back to state S3. No other signal is accepted

within state S4.

- When again in state S3 the system is ready to treat a new "Trigger"signal.
- When the burst ends and the SPS "End of Burst" signal arrives the system jumps back to state S3 where it waits for a new burst.
- If a signal arrives in a state where it is not allowed it can be either ignored or it might cause a transition to an "Exit" state (S5) if the error might compromise the run. In this case the run should be aborted.

4.2.2 Calibration Procedures

Pedestals and LASER calibration routines are transparent from the DAQ point of view. They are performed in the normal way of data taking.

PPG Calibration requires a different procedure, so it will be designed as a separate part of the FSM. The distinction between this procedure and a normal one is made by the mean of a calibration flag "fc" which is set to "1" at the beginning of the run for the PPG.

- The transition from S1 upon the SOB signal will be now to state S2 (Init-Calib). The action performed upon "SOB" signal is a particular case of "act_sob".
- At the end of this action a soft signal "EXTEND2" is called upon which the system (now in state "InitCalib") performs the action "act_ppg_start". The aim of this action is to add a delay between "act_sob" and the generation of the first PPG pulse. This is needed because of the rather long SOB-Busy time of the Event Builder which perturbs the functionality of this routine.
- The "Trigger" signals will now be treated from S2. They call the same function "act_read"
- The soft signal "Next_Event" will call now another action, "act_ppg" which implements the logic for this calibration routine.

The cycle is repeated each burst.

4.3 The Software organisation

The software part of the Local Aquisition implements the Finite State Machine described before and also provides the routines for managing the shared memory, for the dialog with Automatic Run Control, for loading the parameters from the Data Bases etc. It is organized in a set of files like it follows:

- TCL_Daq.c - contains the skeleton of the local aquisition

- TCL_Act.c - contains the routines which implement the "actions"of the FSM
- TCL_Build.c - contains the Data Blocks creation routines
- TCL_CAMAC.c - contains the CAMAC routines
- TCL_Cam_Map.c - contains the routine which sets the CAMAC modules address parameters according to the values set in the Run Control Parameters Database
- TCL_Def.h - contains the definitions of the global variables used by the program
- tiles_creator.c - contains the routines which remove the old memory segment, then create and formate a new one

4.3.1 The Matrix

The skeleton of the Finite State Machine is implemented as a matrix with two fields: states and signals; the elements of this matrix are defined as a structure of "actions" and "next-states".The "actions" are the one performed by the system on the occurance of a particular signal; the "next-states" are the states reached by the system after different transitions.

4.3.2 The Actions

The specific read-out or/and control procedures are done by the mean of those "actions" as it follows:

- Action_sor - It is an action that executes the CAMAC initialisation procedures, it writes the trigger mask on the Output Register and it initialises the calibrations constants.It also writes the SoR Data Block which contains (for TileCal) the Run Control Parameters (from the database)
- Action_sob - It writes the SoB Data Block(which is empty for TileCal), it clears the interrupt channel for the SoB signal and, if the calibration flag "fc" is true, it starts the PPG Calibration routine.
- Action_read - It reads data at CAMAC Level and writes the Event Data Block. Then puts the Next_Event soft signal.
- Action_check_state - Clears and reenables the trigger interrupt channel
- Action_eob - Writes the EoB Data Block (which is empty for TileCal) and clears the EoB interrupt channel.

When the calibration flag "fc" is set, a PPG run will be performed. The specific actions for this type of run are:

- Action_sob - Sets the first PPG Voltage Step calling the routine "ppg_step()" and then puts the EXTEND2 soft signal
- Action_ppg_start - Is an action called by the soft signal EXTEND2. It calls "vos_macros" routines forcing the local system to wait for a time "tau" till the whole system is ready for this procedure. Then it calls the routine "ppg_pulse(channel, module, location)" which will generate the specific voltage pulse to be injected to the ADC inputs selected by "channel" and "module". Simultaneously, a PPG_Trigger signal will be generated.
- Action_read - Is the same action as for normal procedures and it is performed when a trigger signal is accepted; it puts the Next_Event soft signal
- Action_ppg - Is an action called by a Next_Event soft signal when fc is set to "1". It chooses the next PPG Voltage Step and the next group of 10 ADC inputs to be pulsed at different moments and sends the "ppg_pulse" command. Then it clears and reenables the trigger interrupt channel. A new trigger will be generated and treated
- The whole cycle is repeated at each burst.

At the end of the run an "EoR" action is performed in order to clear the CAMAC Branch, to close the memory segment and to release the memory pages used for CAMAC control and auxiliary procedures.

4.3.3 Data Format

Data have a standard format for the whole experiment. This is a tree like structure consisting of a Fixed Header, a User Header and the Blocks of Data, as in fig. 8.

For the Event Builder it is defined a Fixed Event Header, a User Event Header and the Data Blocks are the detectors.

For each detector there is a Fixed Event Header with a structure identical to the one of the Full Event, a specific User Header and the particular blocks of data.

The information about the full event is available on RD-13 Internal Notes; we shall refer now only to the TileCal subevent format.

The Fixed Event Header contains general information about the event:

- Marker - A check word used by the EVT_open routine to check if the event structure exists

- Block Structure Size - Sub-event Structure Size (set by the EVT_close routine)
- Event Type - SoR, SoB, Trigger, EoB
- Run Number
- Spill Number
- Event Number within Run
- Event Number within Spill
- UEH pointer - A pointer to the User Event Header (the relative address)
- CNTS pointer - A pointer to the Event Contents Block
- Error code - It is initialised to "0" by the EVT_init routine

The User Event Header contains particular information for the specific detector:

- UEH Size - The size of the User Event Header
- Time
- Date
- Trigger Type - Physics, Pedestals In-Burst, Pedestals Out-Burst, Pedestals Special, PPG Calibration, LASER In-Burst, LASER Out-Burst, LASER Special
- Event Marker - Check word used by the Online and Offline data analysis programes
- Table Position 0, 1, 2, 3 - The four coordinates of the table position
- PPG HV - The PPG High Voltage Step
- LASER 0, 1, 2 - Three words read from the CAMAC Pattern Unit
- HV Status 0, 1, 2, 3 - Four words for the Photomultiplier Tubes High Voltage Status
- PM Block - The Number of the ADC inputs block which were pulsed for the event recorded
- Three more words reserved for future applications

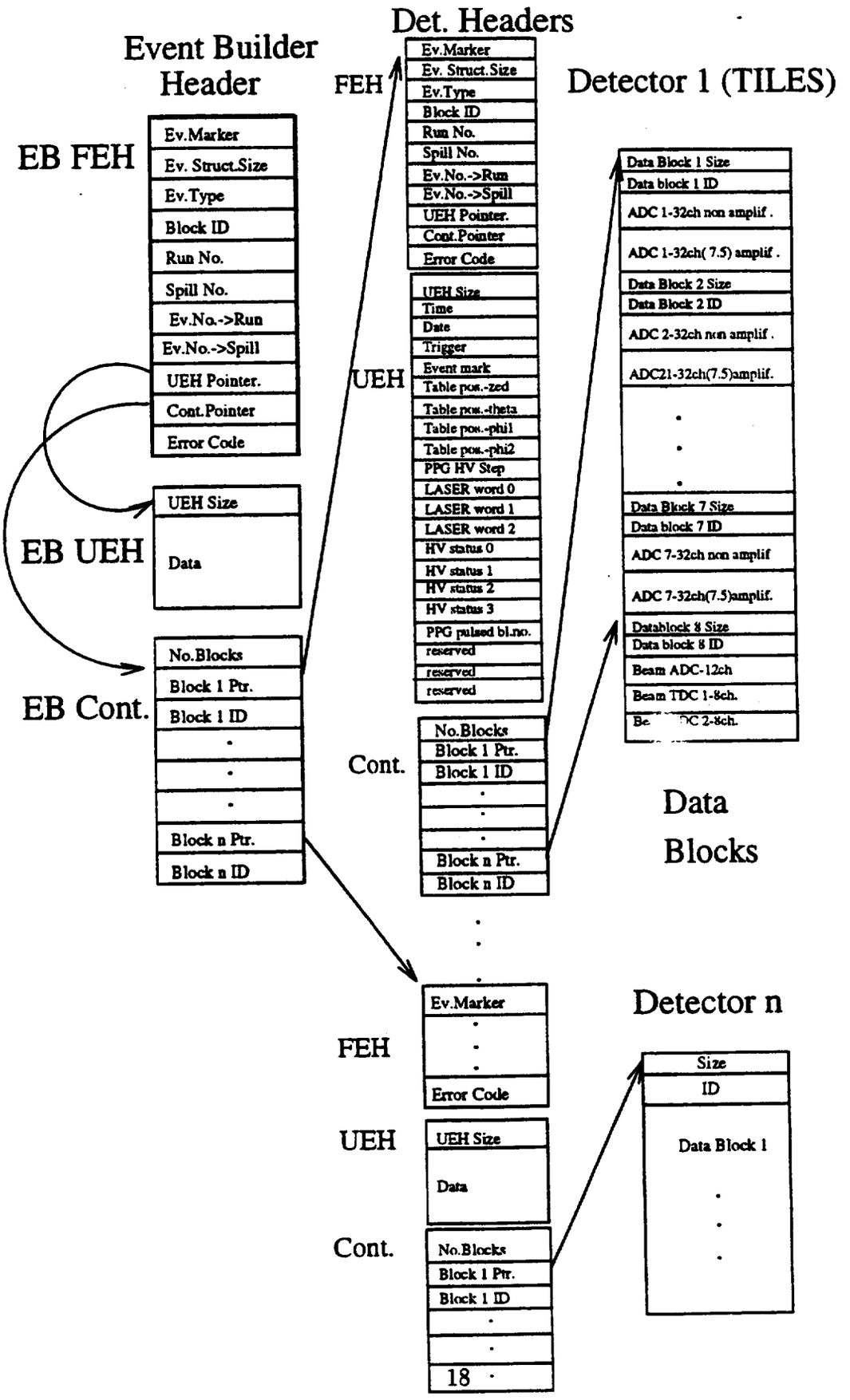

Figure 8: Event Structure

The Contents pointer gives the relative address of a table containing the number of data blocks, their relative address and their identification marker.

For the Tile Calorimeter there are eight blocks of data organized as it follows:

- The first seven ones contain data from each detector ADC module. Each of them has 64 (12 bits length) words corresponding to the 32 channels amplified and non amplified. For each such a block the first 32 words represent the nonamplified paths ant the next 32 ones are the amplified.
- The eighth block contains data from the Beam Elements:

  - 12 words (10 bits) from the Beam ADC ; only the first three words are significant, giving the response of the three beam counters.

  - 28 words from the TDC-s ; the first 5 words of each group of 8 giving the response of the beam chambers.

The correct data format is given by the routines "buildSob", "buildEob", "buildSor" and "buildEvent". These call RD-13 library functions (like EVF_initEvent, EVF_closeEvent, EVF_insertDataBlocks etc.).

The parameters contained by the Fixed Event Header are provided by the "build" routines.

4.3.4 Interrupt management

The local DAQ system is able to wait for and to treat multiple asynchronous interrupts generated from VME level or other sources. The mean to synchronise the user code with the activation of an interrupt service routine is a kernel semaphore. The current version of the EP/LX system allows to wait only on a single kernel semaphore.

The system functionality is characterised by the following aspects:

- A process may attach one or more interrupt sources, thus defining a pool of interrupt sources the process declares to be interested in.
- A process may wait on the sum of the interrupt sources.
- Single interrupt sources may be added to or removed from the set of interrupts wanted
- A process may specify to each interrupt source a user function to be activated
- Single interrupt source operations are provided like: "set", "clear", "test-and-clear" etc.

The RD-34 local aquisition uses RD-13 Multiple Interrupts Libraries: "rd13liberrs.h", "MultiIntFlags.h", "VME_trig.h".

The interrupts initialisation is done inside the "initTileCal" routine. The interrupts are generated by the occurance of SPS Start-of-Burst and End-of-Burst signals, of the Trigger signals or of the so called "soft signals" (as Next-Event). First a function "RD13_flags_initialise()" is called which sets the internal data structures to an initial state. After this, the interrupt vectors are added to the pool of interrupts by the mean of the function "RD13_flags_add()".

Once in state OutBurst (after receiving SOR signal), the system waits for the interrupts by the mean of the library function "RD13_wait_signal()" inside the "waitTileCal()" routine.

The interrupt channels are cleared and reenabled at the end of the routines "act_–" called by the intterrupt signals, by the mean of RD-13 library functions "TM_clear()" and "TM_crW()".

When the routine "exitTileCal()" is called in "act_eor()" at the end of the run, the interrupt channels are cleared and the flags are removed with the function "RD13_flags_remove()".Now the interrupt channels used before are inactive.

4.3.5 Memory management

It is designed by RD-13 who provided also the appropriate routines.
The local memory layout has several purposes:

- Management of the shared memory in the VME/VIC system. This allows creation, mapping, formation, dumping and deletion of a shared memory segment from a Local DAQ.
- Buffer management - The Local DAQ needs to get the addresses of free Event Buffers and to put a mark "filled" once it wrote them. The General DAQ needs to get the addresses of such "filled" buffers and to mark them as "free" as soon as it read them. Apart from the Event buffers there are 3 more special buffers : SoR, SoB and EoB.
- Common access from both the Local and the General DAQ-s to a number of fields which contain for example the Run number, the Spill number, the Event Number etc

The Local DAQ memory is physically contiguous, VME accessible and it is divided into five main parts, as in fig. 9:

- A fixed part containing:
  - Marker - used to check whether the memory has been formatted or not
  - Pointers to other memory areas
  - Fields associated to the local memory area

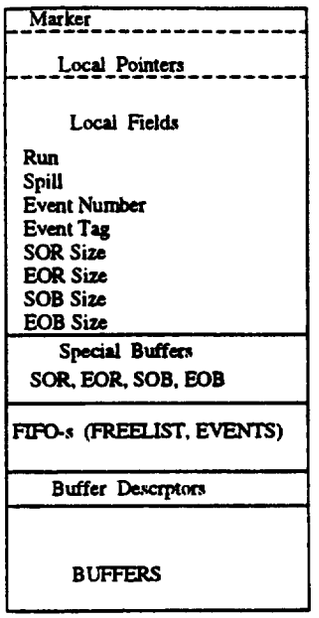

Figure 9: The Memory layout

- The buffers for the DAQ events (SOR, EOR, SOB, EOB)
- FIFO-s - used to maintain the ordered list of free and filled events
- Buffer descriptors - which contain information associated to an event copied into the buffer and the buffer address
- Buffers of fixed size, each containing a physical event

The Local DAQ memory is managed by a set of four operations:

- Creation and removal of a segment

  -They are performed by the mean of the library functions "EBL_createSegment()" and "EBL_deleteSegment" in the program "tiles_creator.c". When a segment is created, a name is associated to it and its base address is returned.

- Map of a segment

  -Is executed by the library functions "EBL_mapSegment()" in the "init-TileCal()" routine of the "TCL_DAQ.c" program.

- Formatting of a segment

  -Is performed by the "EBL_initEvtBuf()" library function in "tiles_creator.c"

- Dump of a segment contents

  -It is executed by "dumpev()" in "do_event()" routine of program "TCL_DAQ.c".It dumps on standard output the contents of a formatted memory segment mapped at address "base".

The buffer management assumes two kind of operations : one to access the free buffers by the Local DAQ and the other to access the filled buffers by the Global DAQ.

The Local DAQ performs the following set of operations:

- Prepares the pointers used by the buffer manager calls.The library function for this operation is "EBL_flushEvtBuf()"and it is called into the "initTileCal" routine.
- Returns a pointer to the next free event descriptor by the mean of the function"EBL_getEvtSlot()" in "do_event()" routine ("TCL_DAQ.c").
- Puts the buffer associated with the descriptor into the EVENTS list and sets the state of the buffer to "available". It is executed by "EBL_sendEvent()" in "do_event".("TCL_DAQ.c").

The Local DAQ reads the value of the single fields by the mean of the library routines "EBL_getTriggerCounters()" and "EBL_getSpillCounters()".

4.3.6 Event Format routines and libraries

- SOR Data Block - Contains all the parameters loaded from the Run Control Parameters Database. It is inserted by a routine "buildSor()" called inside "act_sor()".
- SOB Data Block - Is an empty block for our Local DAQ. It is inserted by a routine "buildSob()" called by "do_sob()" inside "act_sob()".
- EOB Data Block - Is an empty block for our Local DAQ. It is inserted by a routine "buildEob()" called by "do_eob()" inside "act_eob()".
- Trigger Event Data Block - Contains data associated with the detector response for physics or calibration events. This block is organized in standard format by the routine "buildEvent()" called by "do_event()" during "act_read()".

The "act_-()" routines are contained by the "TCL_Act.c" file; "build-()" routines are in the "TCL_Build.c" file; "do_-()" routines are contained by the "TCL_DAQ.c" file.

For each such data block a set of library rotines are called:

- EVF_initEvent() - It is an RD_13 library function which initialises the Fixed Event Header(FEH), allocates the User Event Header (UEH) and allocates the event contents space for all (sub-)detectors. This space will contain for each (sub-)detector the pointer to the relative (sub-)block structure and the identifier. It is called once per event inside "build–()" routine.

- EVF_insertDataBlocks() - It is a library function which requests space for a certain (sub-)detector data block and allocates sub-block space for data read from the physical memory. It also stores the starting point address of the data sub-block so that the structuring process will fill raw data from this starting point. It is called inside "build–()" routine.

- EVF_closeEvent() - It is a library function called at the end of the event formating process (always inside "build–()"routines) to close the structure and evaluate the end point address.

4.3.7 CAMAC routines

The file "TCL_CAMAC.c" is a library of routines designed to provide a user interface for the CAMAC modules used by the TileCal readout. To access the CAMAC level from the VME level, they use the RD_13 library "camaclib.h".

- init_CAMAC - Performs the CAMAC initialisation during "act_sor".

- clear_CAMAC - Clears the CAMAC crates 1 and 2 during "act_eor".

- ccrls() - Clears the CAMAC memory page at VME level.

- write_outr(doutr) - Writes the Trigger Mask value "doutr" on the Output Register during "act_sor".

- cam_trg() - Reads the trigger type from the Event Pattern Unit. It is called in the routine "do_event()" before inserting the specific data block.

- wait_LAM(ncaen) - Checks the value of the LAM signal on "ncaen" detector ADC modules. (The LAM signal marks the end of conversion.) It is called in "buildEvent()" before inserting the specific data blocks

- read_caen(k, subdet_p) - Is the routine which reads data from the detector ADC number "k" and writes them into the specific data block from the starting point "subdet_p". It is called in "buildEvent()" during writing the data block

- clear_caen(k) - Is a routine which clears the detector ADC module no. "k"after reading. It is called immediately after "read_caen()".

- read_lecroy(subdet_p) - Is a routine which reads data from the beam counters ADC and writes them into the specific data block from the starting point "subdet_p". It is called in "buildEvent()" during writing the data block

- clear_lecroy() - Is a routine which clears the beam counters ADC module after reading. It is called immediately after "read_lecroy()".

- read_tdc(k, subdet_p) - Is a routine which reads data from the beam chamber TDC no."k" and writes them into the specific data block from the starting point "subdet_p". It is called in "buildEvent()" during writing the data block

- clear_tdc(k) - Is a routine which clears the beam chambers TDC module no. "k" after reading. It is called immediately after "read_tdc()".

- read_laspatt(r) - Is a routine which reads data from the LASER Pattern Unit no."r". It is called in "buildEvent()" before writing the data block

- clear_laspatt(r) - Is a routine which clears the LASER Pattern Unit module no."r" after reading. It is called immediately after "read_laspatt()".

4.3.8 PPG Calibration routines

- init_ppg(loc) - Sets the PPG hardware to a defined state. "loc" is the CAMAC Station No. for the first PPG module. It is performed during "act_sor()".

- ppg_step(step) - Sets the next PPG voltage step to the value "step". It is performed in "act_sob()" where it loads the value of the first step for the first burst of the run, or keeps the last value from the previous burst. It is also called within "act_ppg()" where it loads the new value of "step" which is incremented after reaching the maximal number of PPG channels.

- ppg_pulse(chann, ppg_mod, loc) - Sends the command to the PPG module No. "ppg_mod", whose CAMAC Station No. is "loc + ppg_mod", for sending a pulse via the channel number "chann". After each call of this routine, "chann" is incremented.

4.3.9 Control routines

At the end of the TCL_DAQ.c file there is a Control Interface which contains a routine for treating the messages coming from the Run Control system ("controlHandler()") and a routine which opens the link and declares the Local DAQ to the Run Control ("controlSetup()").(For details concerning the library functions used by these routines, please see RD-13 notes.)

4.3.10 Database Link routines

The Full DAQ System uses 3 QUID databases:

- A Hardware Database - which declares to the system all the hardware devices involved in the aquisition process
- A Software Database - which implements descriptions of the DAQ configuration in terms of DAQ modules. It declares to the system all the software processes which have to be started at the beginning of the run for the General Aquisition and for the Local Aquisitions and the machines on which they run
- A Run Control Parameters Database - which contains all the parameters whose values might need to be changed for a particular run for each subdetector, parameters as Detector ID-s, Memory Size or Event Size which have to be signaled to the General DAQ and some parameters of general interest(as the Run Number etc.) whose values are supposed to change from a run to another

The Local Aquisitions deal only with the Run Control Parameters Database. For our detector in particular all the parameters from the Run Control Database are read at the beginning of the run and wrote in the SoR Data Block as an array ("rcdb_params[j]") by the mean of the routine "Rcdb_array()". Two files are included for this purpose: "Read_Rcdb.h" and "Rcdb_array.h". The parameters of local interest for TILECAL are read from the Run Control Database by the mean of the routine "map_CAMAC()" (file TCL_Cam_Map.c).

5 The interactive environement - Run Control Facilities

The components of the full Aquisition System are implemented as Finite State Automata (as it was allready mentioned). They are controlled by a Run-Control program which interprets and sends the commands to these components. The Run-Control system was designed and implemented as an interactive environement by the RD-13 group, using ISIS which runs on the UNIX worksations.

The user can operate by the mean of a set of run control windows which appear when the Run Control environement is called.Those are:

- rcl Command window - which displays messages from the Run Control regarding the processes running at that moment
- rcl Processes window - which displays the list of the processes running. Here the user will select the application needed - commonly the run-control one;

a specific window will appear showing the current state of the application and the menu of permitted commands which can be executed within that state.

- DAQ Configuration Run Control window - appears when select the run-control application from rcl Proccesses window.

The possible states of the system are:

- Initial - from which one can select the Hardware and the Software Configuration Databases or can setup(going to state Setup)
- Setup - from which one can configure (going forward to state Configured) or reset(comming back to Initial)
- Configured - where one can start the run (going to state Running) or can abort (comming back to Setup)
- Running - from which the run can be stopped (comming back to state Configured) or paused (going to state Paused)
- Paused - from which one can continue the run (going to state Running) or stop it (going to Configured)

The commands are sent by sellecting one from the menu and sending it by clicking on the SEND button in the window.

From state Configured the user should call the dfe View window which displays all the aquisition processes running (the Readout, the Sampler, the Recorder, the Local Aquisition processes (for TILECAL it is TILES_DAQ) and the monitoring applications.

When the user starts the run (sending the command "run" from state Configured, first will pop up the Run Control Parameters window from which the user can operate on the run parameters from the Run Control Parameters Database. General interest parameters can be set from the main window; the ones specific for each detector can be changed clicking on "Show Detector Parameters" and here selecting the appropriate field. One can also modify the local readout configuration enabling or disabling readout modules in "Show Detector Readout".

The user has also the possibility to call different monitoring applications as the Event Dump (which is very useful when one wants to check out the structure of the data blocks) or the Status Display (which shows useful information concerning the run as Run Number, Spill Number, Number of events FIFO rates, Buffer Occupancy etc).

For a detail description of the Run Control System please reffer to RD-13 Internal Notes No.3, 44, 60 and 69).

6 Run Control Parameters for TILECAL

To be able to start a run it is essential to set correctly the run control parameters for the detector. As it was allready mentioned in the previous section, this can be done from the Run Control Parameters window (fig. 10).

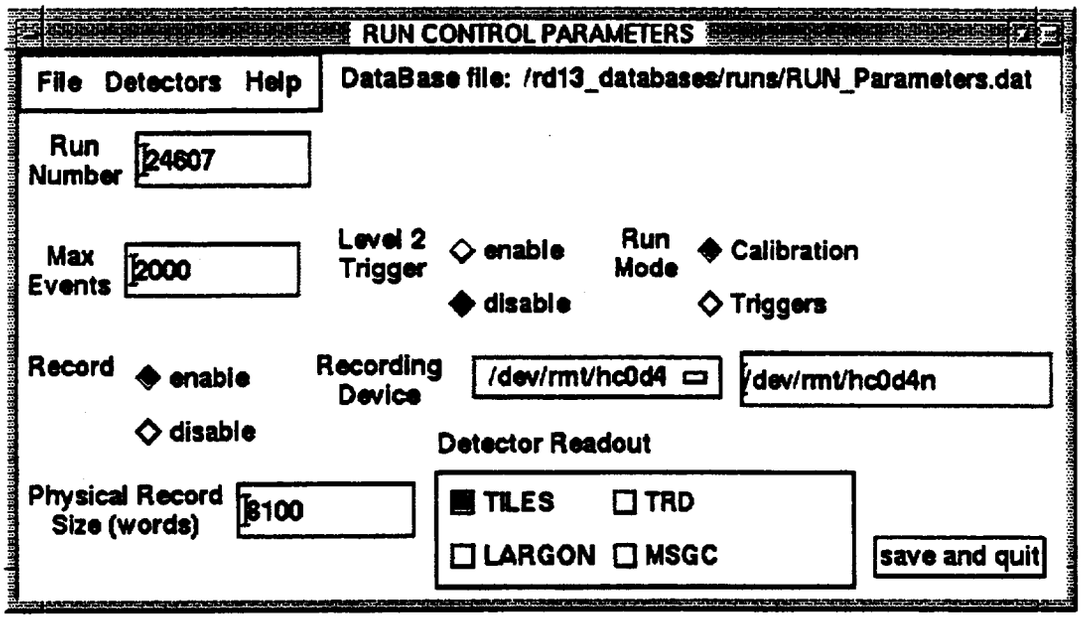

Figure 10: Run Control Parameters window

In the main window the operator should pay attention to:

- Detector Readout - should have TILES enabled (the button pressed in)
- The Run Number - it is set automatically; one should set it by hand only at the the beginning of a new Test Beam period (for a correct initialisation)
- The Physical Record Size should be set just once for a given configuration and it should be at least 4500 for TILES only (assuming that TILECAL tests only 5 detector modules
- The Max Events Number should be set to the appropriate value for the specific run
- The Record Mode should be Enable if one wants to store data on tape and Disable in any other case
- The Record Device should be Correctly chosen (actually /dev/rmt/hc0d4n)
- The Run Mode - should be Triggers when the run is for Physiscs data and Calibration in any other case

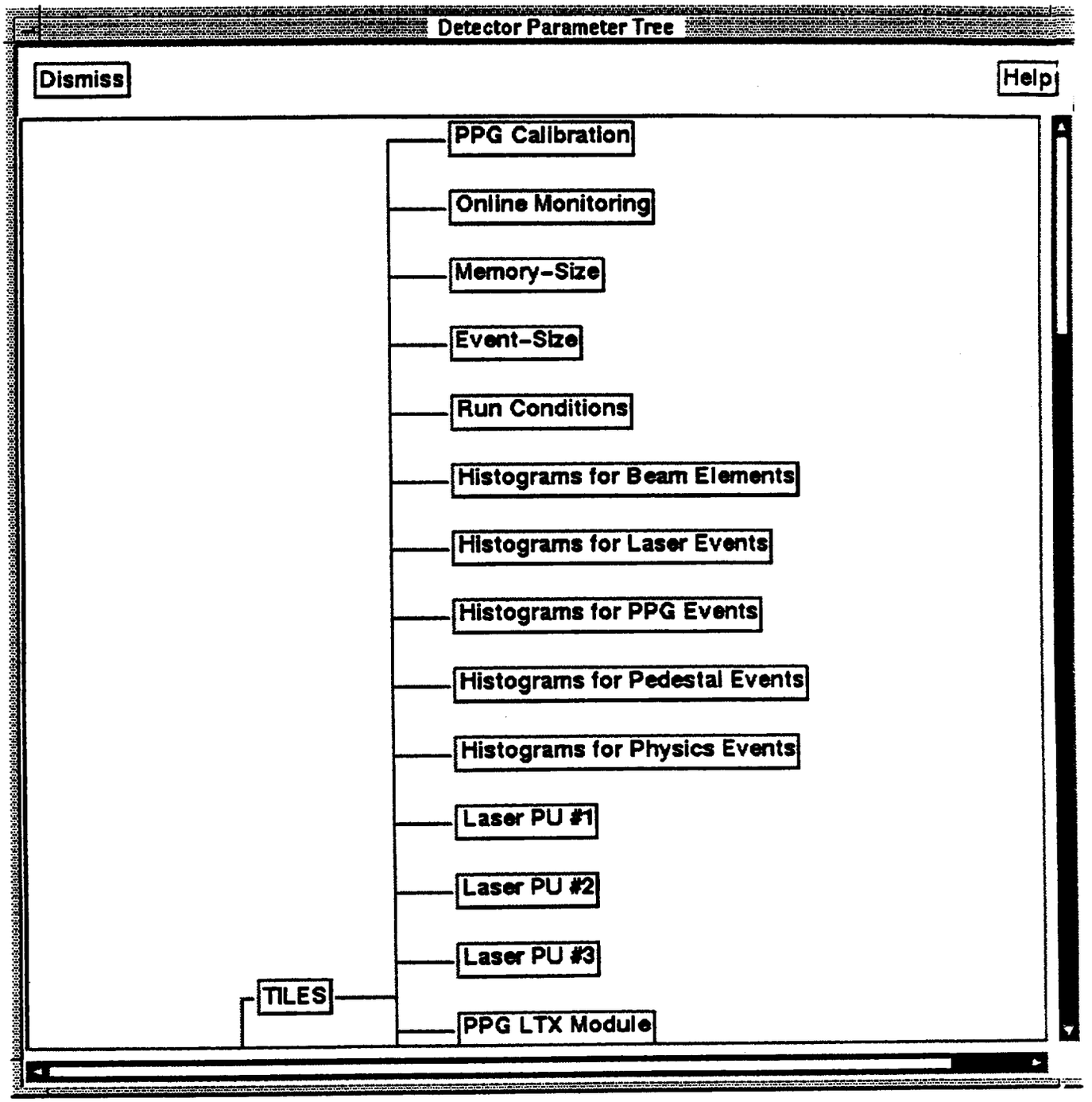

Figure 11: Detector Parameters Tree window

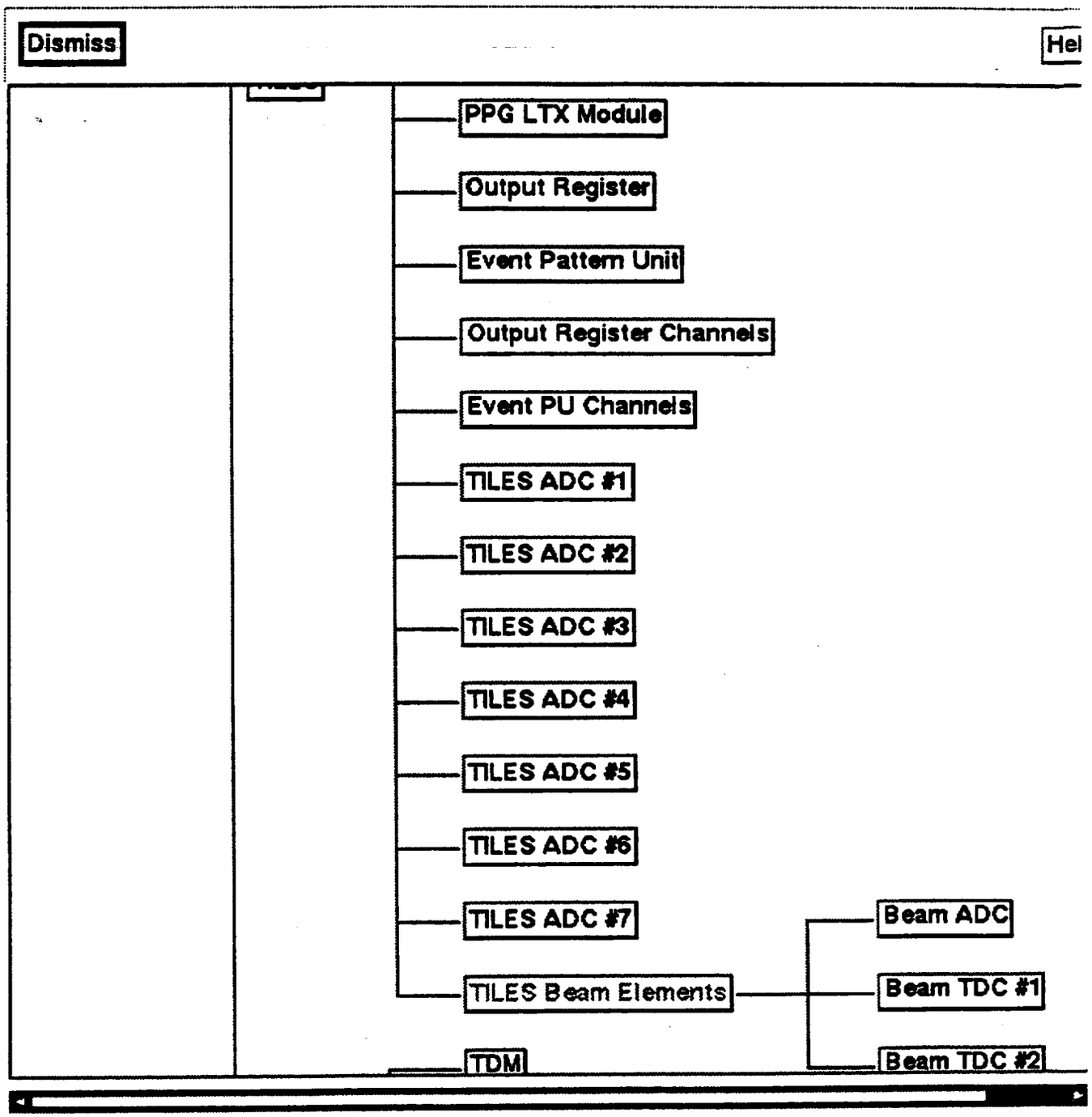

Figure 12: Detector Parameters Tree window (cont.)

For setting the Run Parameters which concern only TILECAL detector one should click on Detectors and select Show Detector Parameters.The Detector Parameter Tree window will pop up. Here it is shown the tree structure of the whole EXPERIMENT which includes TILES as a branch. The user should operate only on the leaves of the TILES branch. The fields commonly used during the runs are Run Conditions, Online Monitoring and Histograms ** fields.

In Run Conditions the operator can choose a particular Trigger Mask for the run. The Physics Trigger is allways enabled !

- When performing a Physics run, the Run Mode should have been set to Triggers in the main window and one can add auxiliary triggers as Pedestal Inside Burst, Pedestal Outside Burst, LASER Inside Burst or LASER Outside Burst.

- Pedestal Special Run, LASER Special Run and PPG Calibration Run are considered as calibrations. For these ones Run Mode should be Calibration

In Online Monitoring one can select the nature of the events needed to be sampled by the Online Monitor and some options (as applying calibrations).

In the Histograms ** fields one can set the parameters of the histograms issued by the Online Monitor for different types of events.

The other fields are not supposed to be changed but only when a new configuration has to be set up !

Memory Size contains the Memory Segment size and it is (for the present configuration) 4194304.

Event size should be (for the present configuration) 4500.

The following fields:

- LASER PU #1
- LASER PU #2
- LASER PU #3
- PPG LTX Module
- Output Register
- Event Pattern Unit
- Beam ADC
- Beam TDC#1
- Beam TDC#2

contain information about the CAMAC Crate and Station numbers of the CAMAC modules, their number of channels and the status(Enable-Disable).

The fields TILES ADC #1 - #7 contain the CAMAC Crate and Station numbers of the detector ADC CAMAC modules and their number of channels.

The Output Register Channels field and the Event Pattern Unit Channels field give the correspondance between the bit number of the module and the Trigger Mask and the Trigger Path respectively.

The PPG Calibration field contains the values of the parameters for the PPG Calibration procedure:

- The Starting HV step - should not be less than 2
- The number of total HV steps - up to 14 when starting from 2
- The number of PPG Modules - actually 3
- The first PPG module CAMAC Station number in the remote controlled CAMAC crate - actually 18

Using different combinations in the values of the number of PPG Modules and the first PPG module CAMAC Station number one can access each PPG module independently.

If one clicks in the Run Control Parameters window on Detectors and chooses Show Detector Readout, a new window (Detector Readout Tree) will pop up (fig. 13). Here one can enable or disable the TILES detector ADC modules or the Beam Elements block.One should notice that these correspond to the blocks of data in the Event Structure. Disabling one of these modules will not take out a data block from the structure but this one will be filled with zeros !

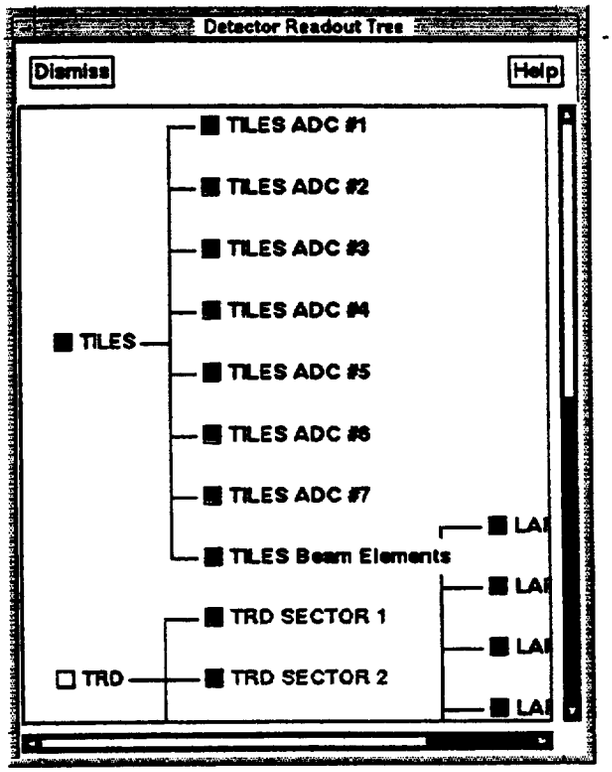

Figure 13: Detector Readout Tree window

7 Parameters and performances

The Local DAQ system is fully secure against spurious signals or fake triggers due to a fully protected Trigger and Busy Logic and Interrupt Control.

The accuracy of data is given by the detector ADC-s performances.So:

- All signal paths are adapted on 50 ohms (NIM standard).
- The full scale input charge is 112.5pC (amplified) and 900pC (non-amplified)
- The conversion gain is 4counts/pC (non-amplified) and 30counts/pC (amplified)
- The integral non linearity is +/- 2.5 counts (non-amplified) and +/- 6.5 counts (amplified)
- The integration time (gate width) was choose to be 300ns

The maximum triggger rate accepted by the full system (meaning the General Aquisition and our Local Aquisition alone) was 820 triggers for the 4.5

seconds of the extended burst, which is about 520 for the 2.5 sec. when particles arrive and 300 for the two extensions. The main limitation in speed is due to the latency of the CAMAC readout. The average dead time is 4.5 msec and most of it is spent for reading the 64*7 words of the detector ADC-s.

8 Contributions and references

I am indebted to Dr. Giuseppe Mornacchi, spokesperson of the RD-13 collaboration, for allowing us to present here some information about the General DAQ. Special thanks also for him and his colleagues for the support they provided to us in the design and integration of our Local DAQ.