Chapter 4 : Readout, Trigger and DAQ

Introduction

The main purpose of the readout, trigger and data acquisition system is to convert the analogue outputs of the PMTs into a readable input for the off-line reconstruction software. A schematic view of the system is shown in figure 4.1.

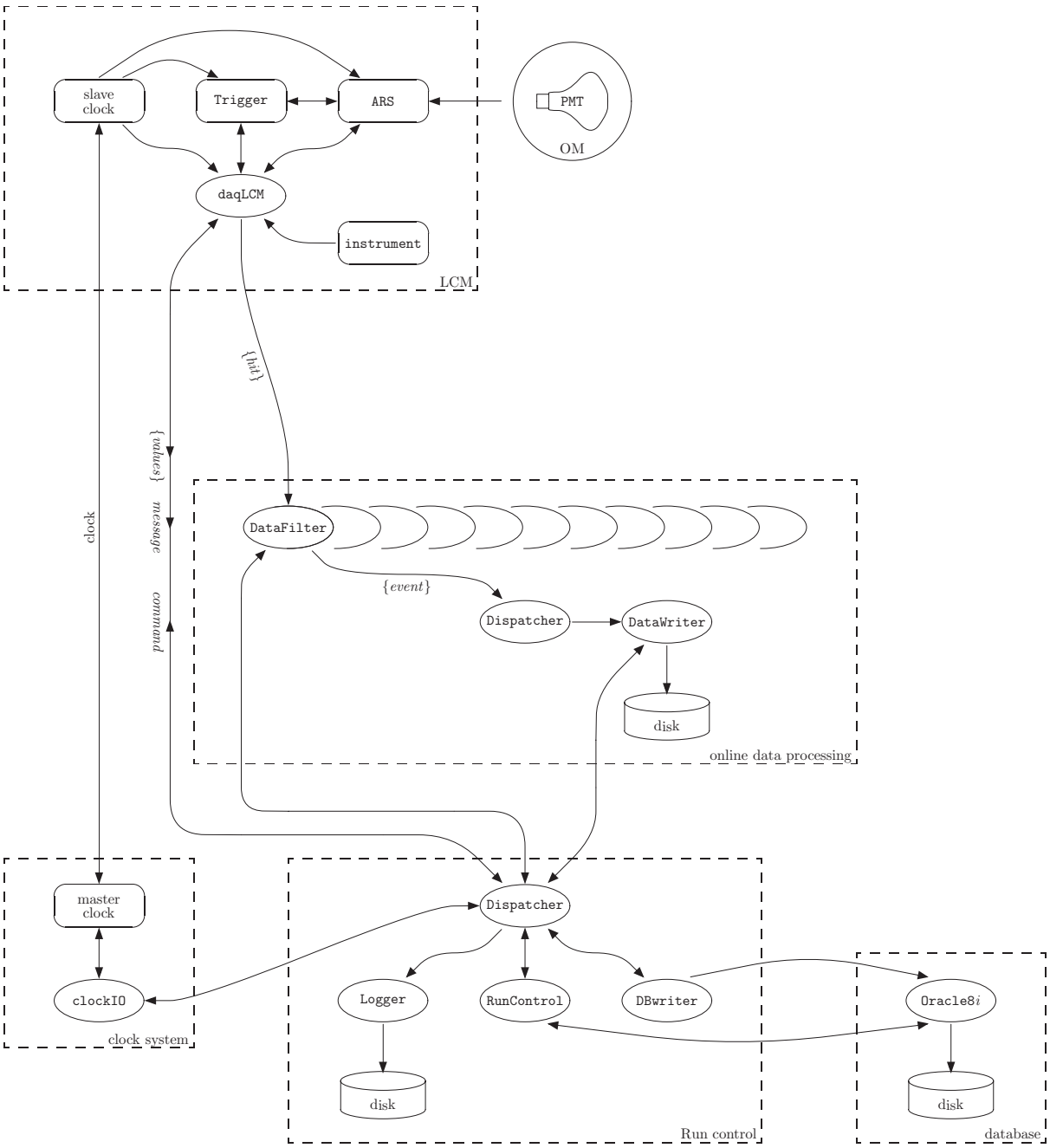

Figure 4.1: *Schematic view of the data acquisition system.*

The ellipses correspond to processes and the boxes to hardware devices. The links between processes indicate the exchange of information (commands, data, messages, etc.). The connections with hardware devices inside the LCM are made using electrical signals. The readout system consists of ARSs which digitise the charge and the time of the (accepted) analogue signals of the

PMT. The combined data from the PMT are generally referred to as a hit; it can be a single photo-electron (SPE) hit or a complete waveform (WF). The arrival time is determined from the signal of the clock system in the LCM. The on-shore clock system (master) drives the clock system in the LCMs (slave). The data produced by the readout system are collected by the processors in the LCMs and transferred as an array of hits to the on-shore data processing system. The length of these arrays is determined by a predefined time frame of 10-20 ms and the singles rates of the PMTs. All data corresponding to the same time frame are sent to a single processor. The on-shore data processing system consists of a limited number (< 100) of processors. With this system, the (physics) events are filtered from the data using a fast algorithm. The initial data flow can be reduced by the off-shore trigger system. The data from the readout of the hardware devices (e.g. instruments) in the LCM (or OM) are transferred as an array of sets of parameter values. The run control system allows the state of the system to be modified. The database system keeps track of the history of the detector and the data taking and is also used for storing and retrieving configuration parameters of the whole system. It is planned to have a high bandwidth data link from the data storage system to the outside world. Through this link, the output data can be sent real time to any (European) computer centre. This provides worldwide fast access to the data.

DAQ model

The processes in the DAQ system can be considered as a set of concurrent state machines. The possible states and the foreseen transitions are shown in figure 4.2. In general, transitions are caused by events which are generated by the operator of the **RunControl** programme. In some cases, they could be generated by an error condition during configuration or at runtime. These are not shown in figure 4.2.

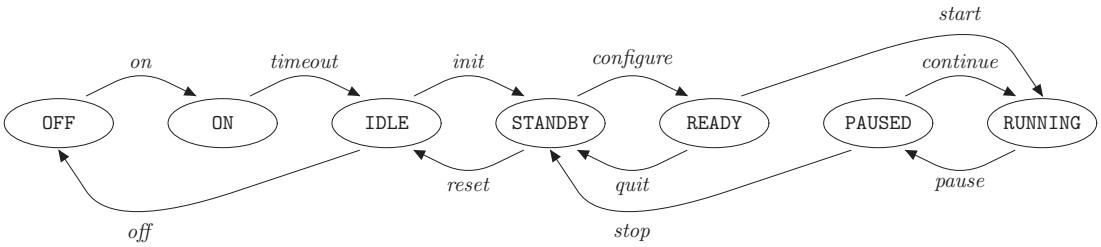

Figure 4.2: *States and transitions of the DAQ system*

The actions which are needed to make a transition between two states in the DAQ model can be considered as procedures. These procedures are different for the various processes (and hardware devices). The power *on* procedure is described in the Power Section in the Infrastructure chapter. The **ON** state is a temporary state of the String Power Module (SPM); after a fixed time delay (around 10 min.), the sectors in the corresponding line are powered according to the last (hardware) settings of the SPM. Other settings can be applied when slow control access from the String Control Module (SCM) to the SPM is made before the timeout event. The (hardware) settings can be overwritten any time after successful start up of the SCM. After successful completion of the power *on* procedure, all hardware devices enter the state **IDLE**; the off-shore processors are booted. The *init* event initiates the launching of the main processes necessary for the data acquisition. The hardware devices and the software enter the state **STANDBY**. The human operator can then choose the mode of data taking. Two basic modes can be distinguished, namely time calibration data taking and physics data taking. The *configure* event causes the hardware and the software to be initialised accordingly. The configuration procedure of the detector consists of initialising all hardware devices which are used for the data taking. For this, a (virtual) description of the complete detector setup is created by the **RunControl** using the detector setup tables of the general database. The device specific configuration data is then sent to the corresponding **daqLCM** processes. Each **daqLCM** process writes these data into the hardware devices via a local bus (RS485) and a universal interface circuitry (UNIV1) [2]. Similarly, all processes are initialised by sending them their running parameters. These parameters are also stored in the general database. In the DAQ model, it is not foreseen to (re)configure the detector in the **PAUSED** and **RUNNING** states. In order to control the off-shore data flow, the *pause* and *continue* commands respectively disable and enable the ARSs in the readout system using the clock system. Every event generated by the operator is stored by the **Runcontrol** programme in the general database. For this, the GPS time is obtained from the clock system through the PCI bus of its host (not shown in figure 4.1) using the IRIGB protocol [9].

Time calibration data taking

For the time calibration data taking, the optical beacons (OBs) are configured such that a light flash is produced when the corresponding clock signal is received. The ARSs in the readout system

and those of the optical beacons are temporarily enabled during the light flash. Each flash produces about 10 kByte of data. The available bandwidth allows up to 10^5 flashes per second. The filtering algorithm of the **DataFilter** programme is by-passed and all (calibration) data are saved on disk via the **DataWriter** programme. The data can be written to tape continuously when flashing 10^3 times per second. The run is closed when all OBs have been fired a (programmed) number of times; the DAQ ends in the state **STANDBY**.

Physics data taking

For physics data taking, the optical beacons are inhibited and the filtering algorithm of the **DataFilter** programme is activated. Optionally, the off-shore trigger system can be enabled to reduce the data flow. The physics data, the PMT calibration data (see section PMT Readout) and the clock calibration data (see section Clock system)are taken simultaneously. After the run *start*, the data taking continues autonomously until a *pause* event is issued by the human operator. In order to archive data efficiently, the **RunControl** programme updates the run number regularly.

PMT readout

The main function of the PMT readout system is to time stamp the (accepted) analogue signal and to digitise the charge. With an amplification factor of 5×10^7, the amplitude of the analogue signal from a single photon electron is about 60 mV on a 50 Ω load. The time digitisation is determined from the external clock signal. The design accuracy is 0.5 ns. The system should handle an average singles rate of 100 kHz and surges of up to 250 kHz due to bioluminescence bursts.

For the implementation of the readout system, an Analogue Ring Sampler (ARS) chip has been developed. It has three analogue inputs, a clock input, a trigger output (L0), two trigger inputs (L1 and L2), a serial IO port and a (fast) output port. The analogue inputs can handle pulses up to 4.5 V. The clock input is used for the time digitisation. The IO port is used for the configuration of the ARS. The (fast) output port is used for the readout of the ARS; it has a bandwidth of 20 Mb/s. The ARS has an internal 24-bit clock register which can be reset with an external reset time stamp (RTS) signal. A time to voltage converter (TVC) is used to interpolate between clock pulses. With a 20 MHz external clock frequency and with 8-bit resolution for the TVC signal, the corresponding time resolution is 50/256 = 0.2 ns with a dynamic range of 840 ms. This is well within the design accuracy. The ARS can discriminate between single photo-electron hits and complete waveforms. The corresponding dynamic ranges of the charge digitisation are 15 p.e. and 20 p.e., respectively.

In the following, a brief description of the ARS functionality is given. A more detailed description can be found in ref. [3]. The anode signal of the PMT is connected to the (main) analogue input of the ARS. This signal is discriminated by a threshold (typically 0.3 p.e.). If accepted, a L0 trigger is generated. The charge of the pulse is then integrated during a programmable time interval (25-80 ns), the internal clock register is stamped, and the Time to Voltage Converter (TVC) is latched. The integrated charge, the time stamp and the TVC are temporarily stored in a mixed analogue-digital pipeline memory. The ARS has a pulse shape discriminator (PSD) which can distinguish a waveform (WF) hit from a single photo-electron (SPE) hit. The WF contains not only the SPE information but also 128 samples of the analogue signal. The sample frequency is programmable between 0.3 - 1 GHz. The pulse shape discrimination is defined by a threshold (2 - 10 p.e.), a time-over-threshold criterion (10 - 50 ns) and/or a double-pulse criterion. The double-pulse criterion is applied within the charge-integration time window. In order to extend the dynamic range of the charge digitisation, the second analogue input is connected to an attenuated anode signal and the third input to one of the dynode outputs of the PMT. Optionally, the shapes of these signals are sampled in addition to that of the anode if a pulse has an amplitude larger than 20 p.e. The dynamic range of the charge integration is then increased to 200 p.e.

The pipeline memory of the ARS can store up to 16 SPE hits or 4 WF hits. For the final digitisation and readout, it has a two-way trigger function. These trigger functions are organised as follows: a label "accepted" is assigned to a hit in the pipeline if either a L1 trigger arrives within a (programmable) time window (8 - 30 ns) after a (programmable) delay (55- 250 ns) or a L2 trigger within a (programmable) window (0.8 - 7 µs) after a (programmable) delay (7 - 36 µs) with respect to the L0 trigger. For each hit with the label accepted, the analogue part (charge, TVC or WF) is digitised with a dual 8-bit Analogue to Digital Converter (ADC). The digitised data are serialised and put on the IO port. When the hit has been digitised completely or when after a fixed latency (37 µs) neither trigger condition is fulfilled, the memory location is made available for subsequent hits. Six types of hits are generated by the ARS. Each contain an 8-bit header with the ARS identifier (2 bits), the hit type (3 bits), a PSD flag, an ACQ/BUS flag, and a counting rate monitor (CRM) flag. A short description of the hit types is given below, and a summary of the corresponding data volumes can be found in table 4.1.

SPE A single photo-electron hit is generated if an input trigger (L1 or L2) coincides with the (internal) L0 trigger of a hit. It contains the header, the integrated charge (1 Byte), the time stamp (3 Bytes), and the TVC. The rate equals the L0 rate reduced by the fraction of data selected by the (upgrading) trigger gate.

WF A waveform hit is generated if the signal passes the PSD criteria. It contains the SPE information and 128 samples of the anode signal. The rate equals the PSD rate reduced by the fraction of data selected by the (upgrading) trigger gate. When both ARS chips are occupied by waveform hits, the data are reduced to those of a SPE hit. The PSD flag in the header is then set correspondingly.

A subset of waveform hits can contain waveform information from dynode signals as well. The rate corresponds to that of signals with a pulse height larger than 20 p.e.

RTS Reset time stamp events are generated each time the internal clock register is reset. This reset occurs when the ARS counter reaches its maximum or when an external RTS signal is applied. The header contains the last time stamp value before the reset. The rate equals that of the RTS signal.

STATUS Status events are generated by the ARS at *power on* or when slow control access starts or terminates. The ACQ/BUS flag in the header indicates the actual status. Status events contain the current time stamp value.

CRM Counting rate monitor events are generated by ARS each time the L0 precount is reached. It contains the header, time stamp and the time needed to reach the precount. The rate equals the L0 rate divided by the precount.

Event Type	data (Bytes)
SPE	6
WF	263
WF + dynode	519
RTS	4
STATUS	4
CRM	5

Table 4.1: Summary of the ARS *hit* data.

The configuration parameters are set from **daqLCM** process through the FPGA and the serial IO port of the ARS. A summary of the configuration parameters is given in table 4.2. All configuration parameters are transmitted as a single frame of 239 bits with a 3-bit header containing a Read/Write option (1 bit) and the ARS identifier (2 bits).

The trigger function of the ARSs can be bypassed (All_t1b parameter in table 4.2). Each accepted signal (L0) is then digitised and readout. In this case, the output data of each ARS are time ordered. The digitisation can be disabled (enabled) through the clock system. This allows control of the off-shore data flow. For the PMT calibration, each ARS can generate a single pulse or a train of pulses synchronous with the reset time stamp signal. These pulses are used to drive the LED that illuminates the photocathode of the PMT. In order to minimise dead time, two ARSs are used per PMT. They are arranged in a so-called token ring. With a singles rate of 70 kHz, a 2% fraction of WF and a << 1% fraction of WF + Dynode events, the (maximal) data rate is about 7 Mb/s per PMT. This data rate is shared between the (fast) output ports of the two ARSs and is well within the bandwidth limit of 20 Mb/s of each ARS. A third ARS is used for triggering purpose only; it is not read out. The three ARSs are mounted on the so-called ARS mother board inside the LCM.

Parameter	Description
Dyn_th	Trigger threshold level of dynode selection
Trig0_th	L0 trigger threshold level
PSD_th	Threshold level of the PSD amplitude criterion
Spe_clk	Integrator cycle period
Spe_gate	Width of the integrator end of gate
Spe_pipe	Reset and write cycles width of the Pipeline
Sel_pled_clk	LED pulse burst rate selection
En_acq	Enables event acquisition, this bit is ANDed to the signal *En_acq*
All_spe	Forces SPE type events
All_wave	Forces Waveform type events
PSD_tot	Value of the PSD pulse width criterion
CRM_pc	Value of the CRM precount (number of event pulses that triggers the CRM)
CRM_sel_clk	Selects the CRM counting frequency
CRM_w	Value of the CRM warning threshold
Burst	Enables the 1024 LED pulse burst mode, single pulse by default
Sel_pulse	Selects the Reset Time Stamp signal to generate LED pulses instead of prescaled clock signal.
En_pulser	Generates one or a burst of 1024 LED pulse(s)
Acc_t2	Acceptance gate width of the L2 trigger requests
Wacc_t2	Pipeline memory wait state time
Delay_t2	Internal delay on the L2 input signal
Delay_t1	Internal delay on the L1 input signal
All_t1b	systematically triggers any event (like L1 triggers)
T0_width	L0 trigger minimum pulse width and maximum pulse width divided by 4

Table 4.2: Summary of the ARS *configuration* parameters.

Trigger

The purpose of the off-shore trigger is to select hits in the ARS pipeline data streams for digitisation, readout and transfer to the online data processing system on shore. This selection is required if the data rate exceeds the capacity of the front-end electronics (ARS), the off-shore data acquisition system, or the on-shore data processing.

The intensity of the optical background, in particular the bioluminescence background, is highly variable on time scales from seconds to months. The trigger system must be capable of adjusting to these variable background conditions in such a way as to retain the maximum of useful data, where useful data include data for background studies as well as signal data from neutrino interactions. The off-shore trigger is constructed using the (accepted) signals from the optical modules of two adjacent storeys within a single string. The trigger conditions are tested in each of the LCMs. If `loose' conditions are satisfied within the LCM, a local L1 trigger is formed. The L1 trigger is sent directly to just the ARSs of the OMs on that storey. This provokes the digitisation and readout of the pulses present within a (predefined) narrow time window. If `tight' trigger conditions are satisfied within the LCM, a L2 trigger is distributed to the full detector and initiates the digitisation and readout of the data in all LCMs within a programmable time window. The exact definitions of the L1 and L2 trigger requirements are selectable via slow control commands. They will be set as loose as possible, depending on the background counting rates and on the capacity of the front-end electronics, the off-shore DAQ, and the on-shore processors. A looser trigger increases the efficiency for signal events, including unexpected signals, and allows a more complete analysis of backgrounds and systematic errors. The most favourable trigger configuration would be no trigger at all. In this case, all data would be sent to shore for analysis.

Implementation

The off-shore trigger is formed in the trigger card of the LCM. It is based around a Field Programmable Gate Array (FPGA) which can be configured by slow control, via the **UNIV1** interface, to select the L1 and L2 trigger criteria. Figure 4.3 shows a general schematic of the trigger implementation. The anode output of each PMT is connected to three ARSs. Two ARSs are used for the readout; their (common) threshold (*th1*) would normally be set to select single photo-electron pulses. The third ARS is used for triggering purposes only; its threshold (*th2*) would normally be set to select pulses corresponding to two (or more) photo-electron pulses.

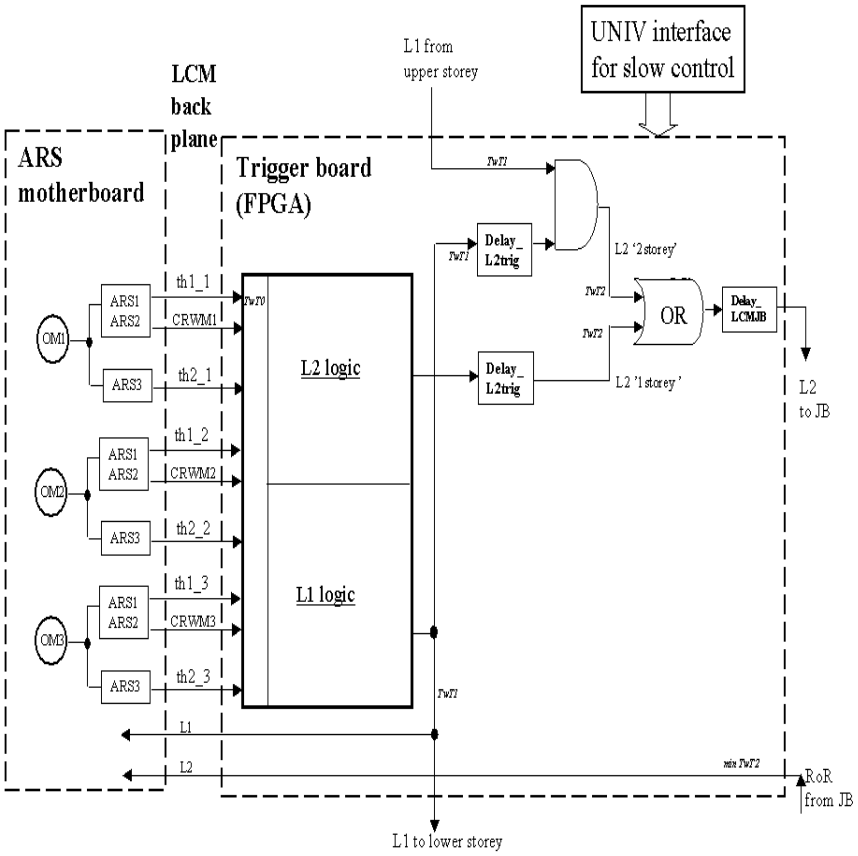

Figure 4.3: *The off-shore trigger scheme.*

The L1 trigger is based on local coincidences within a storey. It initiates a readout of just the local storey; it does not generate a global readout request. It is also passed to a lower storey for use in the formation of the L2 trigger. When constructing the L1, OMs with unusually high singles rates, for example due to bioluminescence, are temporarily removed from consideration if the Counting Rate Monitor warning signal of the ARS is active. The L2 trigger on the other hand does generate a global readout request (RoR). It consists of two components: the L2 '1 storey' trigger is generated using information from the local triplet of OMs in the LCM (this is normally a `tighter' condition than the L1 trigger previously discussed) and the L2 '2 storey' trigger generated by a coincidence between the L1 signals of two adjacent LCMs. The coincidence is formed by delaying the L1 signal of the lower LCM by the time taken for the L1 signal of the upper storey to reach the lower storey (delay-L2trig in table 4.4). The coincidence gate between adjacent LCMs is defined by the width of the pulses (TwT1 in table 4.4) and needs to be large enough (~70 ns) to be efficient for both upward and downward going track topologies.

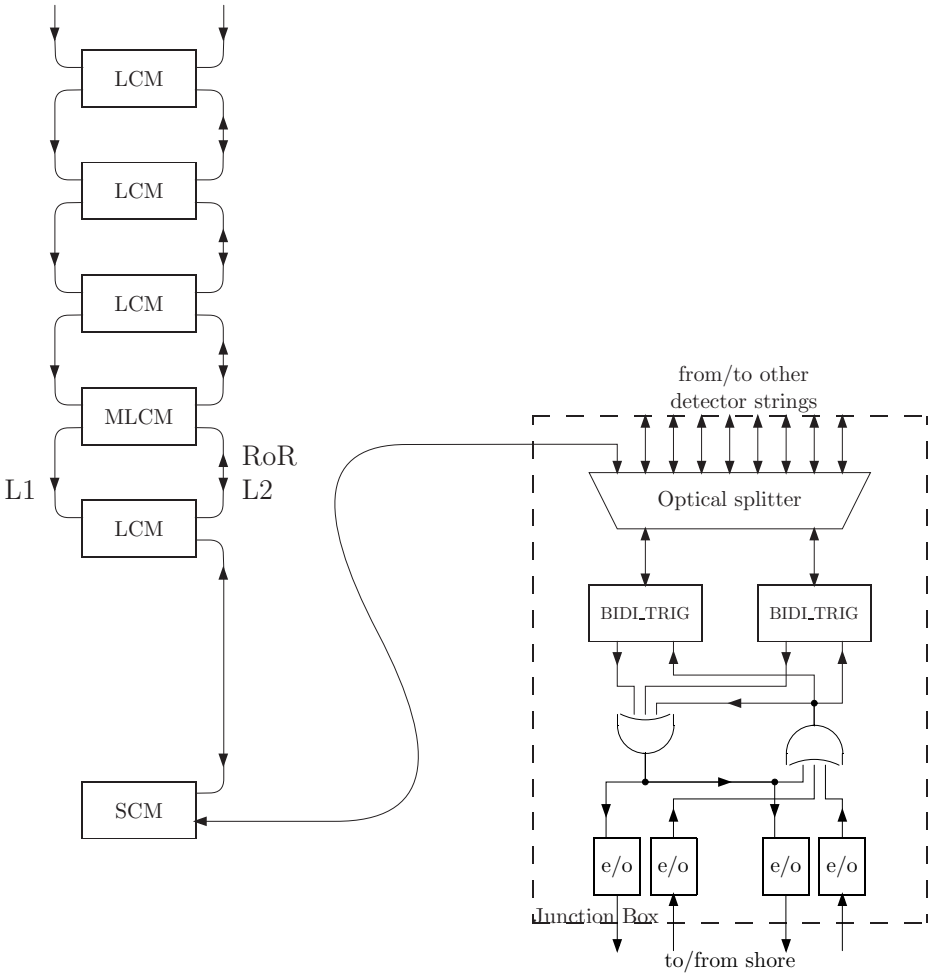

Figure 4.4: *Optical fibre layout for the off-shore trigger.*

The layout of the optical fibres used by the trigger system are illustrated in figure 4.4. The L1 and L2/RoR signals each use a separate optical fibre. Both signals are daisy chained between LCMs. The L1 signal is passed from an upper storey to a lower storey. The L2 signal from a particular storey is passed through all lower storeys to the SCM and from there to the junction box. At the junction box all L2s from all lines are ORed (updating) and the RoR broadcast back along the L2 fibres to all LCMs. The OR at the junction box is performed using a passive 16 x 4 optical splitter. The retransmission of RoR is achieved using the standard BIDI_TRIG unit. Four `spy' fibres are incorporated in the EO cable to bring the RoR signal to shore and allow generation of a 'fake' L2 from shore.

In order that the L2 acceptance window of an ARS be independent of the LCM which initiated the RoR, it is necessary that the arrival time of L2 trigger at the JB be independent of the location of the LCM originating the trigger. This is accomplished by introducing a programmable delay on the L2 trigger signals sent to the Junction Box. This delay depends on the cable length between the LCM and the JB. L2 signals from LCMs with short cable paths to the Junction Box have large delays (up to 3.5 μs), whereas L2 signals from the most distant LCM would have no additional delay at all.

The L2 acceptance time window of each ARS is delayed appropriately for its unique position with respect to the junction box (Delay_t2 in table 4.2). The width of the acceptance window (Acc_t2 in table 4.2) is defined by twice the maximum time taken for a muon to traverse the whole detector (around 4 μs) and is the same for all ARSs. The exact definitions of the L1 and L2_1storey triggers are selectable by slow control. They are based on requiring time coincidences between the various *th1* and *th2* signals of the three OMs within the storey. The currently envisaged possibilities are summarised in table 4.3. It should be noted that many of the options are included for debugging purposes.

Trigger option
L0_1 @ *th1*
L0_2 @ *th1*
L0_3 @ *th1*
1 @ *th1*
1 @ *th2*
2 @ *th1* OR 1 @ *th2*
1 @ *th1* AND 1 @ *th2*
2 @ *th1*
2 @ *th2*
2 @ *th1* AND 1 @ *th2*
3 @ *th1*
3 @ *th2*

Table 4.3 : Trigger options for the L1 and L2_1storey triggers.

The overall reliability of the trigger system is enhanced with the two-level trigger system described here (local L1 and global L2 triggers). A failure in the transmission of the L1 signal at a particular point in the line, would only effect the local formation of the L2 trigger; the local L1 readout of the storey would still occur and RoRs from other storeys would be received. A break somewhere on the L2/RoR fibre would cause loss of L2 triggers for all storeys above the location of the failure; nevertheless the L1 triggers would still be active. To reduce the probability of a single point failure in the broadcast of the RoR at the junction box, a second BIDI_TRIG unit is included on the second output of the optical splitter.

Setting up and monitoring the trigger

A list of the configuration parameters is given in table 4.4. These parameters are set from the **daqLCM** process at *configure*.

Parameter	Description
L1_enable	enable/disable L1
L2_2storey_enable	enable/disable L2 2storey
L2_1storey_enable	enable/disable L2 1storey
L1_config	choose trigger for L1
L2_config	choose trigger for L2_1storey
Delay_L2trig	delay between storeys for L2_2storey
Delay_LCMJB	delay between LCM and JB
TwT1	coincidence window for L2_2storey
TwL1	minimum width of L1
TwT2	minimum width of RoR

Table 4.4 : Parameters of the trigger card to be configured by the slow control.

Successful operation of the off-shore trigger, and also its simulation in Monte Carlo, relies on the

correct setting and knowledge of the various delays and pulse widths involved in its construction. Although they will be calibrated on-shore prior to deployment, the design incorporates facilities to allow measurement and monitoring of these quantities *in situ*.

- Test pulse signals can be generated at the input to the ARSs via the clock distribution. By generating L1s in adjacent storeys and monitoring the L2`2_storey' trigger rate as a function of the set value of delay_L2trig, it is possible to establish the correct setting for this quantity *in situ*.
- By generating L2s using the test pulse facility and measuring the arrival time of the RoR on-shore with the 'spy' fibres it is also possible to measure the relative time delays between storeys. The spy fibre, in conjunction with the appropriate trigger option, also allows measurement of various pulse widths utilised by the trigger card.
- Optionally the DAQ card can also time stamp the time of the L1 and L2(local) triggers and include them in the data flow. This will facilitate the study of the trigger behaviour during dedicated trigger runs.

Trigger and data rates

The following estimates of the expected trigger rates are based on measurements from various site exploration studies [7]. As these studies were performed with 8-inch tubes, the rates have been scaled to the 10-inch tubes. The measurements indicate a baseline rate of 30 - 60 kHz, depending on the period of the measurement. The largest contribution to this background is from ^{40}K decays. In addition, frequent bursts of bioluminescence activity are also observed which increase the singles rate into the MHz range for periods of the order of seconds. This effect is found to be local and does not give rise to significant correlated hits between adjacent storeys. As can be seen from table 4.5, the observed baseline counting rate decreases rapidly at increasing pulse-height thresholds, indicating that the background is dominated by single photon emission.

Threshold (p.e.)	Rate
0.3	60 kHz
1.0	30 kHz
1.5	4 kHz
2.0	0.51 kHz

Table 4.5: Measured baseline counting rates from site exploration tests, for various thresholds, scaled to 10-inch PMTs.

The data volume for a given trigger depends on the fraction of waveform events which is estimated to be 1% plus the fraction of double pulses in the integration gate (50 ns). Considering the optimistic case of a constant singles rate of 60 kHz, the waveform fraction is 1.3% and the data volume per hit is then 0.987 x 48 bits + 0.013 x 2104 bits ~ 75 bits. The total data volume would be 4.5 Mb/s per OM, 68 Mb/s per MLCM, or 405 Mb/s per string. A more realistic estimate should take into account the effect of bioluminescence. Assuming the bioluminescence rates observed in test 1.6, the average singles rate increases from 60 to 75 kHz. Here a cutoff of 500 MHz has been applied on the maximum OM singles rate, close to the output limitation of two ARSs. Including bioluminescence also increases the fraction of waveforms from 1.3 % to 1.4 % leading to unfiltered data rates of 6 Mb/s per OM, 90 Mb/s per MLCM, or 540 Mb/s per string.

The data volume can be reduced in a number of ways; for example, the waveform criteria could be tightened, a lower singles rate cutoff could be introduced, data truncation could be performed etc. Here the possibility of activating the off-shore trigger is considered. The proposed default offshore trigger is configured as follows:

- the L1 trigger is chosen to be '2@ *th1* OR 1@ *th2*', i.e. a single hit above the higher

threshold (*th2*), or two hits above the lower threshold (*th1*) on two different OMs within a +/- 20 ns coincidence gate.

- the L2'1 storey' trigger is chosen to be '1@ *th1* AND 1@ *th2*', i.e. the presence of an asymmetric pair within the +/- 20 ns coincidence gate. (The L2 readout gate is set to 4 μs).
- the L2'2 storey' trigger, by definition, is a coincidence of L1s between adjacent storeys, within a +/- 80 ns coincidence window.

Using this trigger and applying a high threshold (*th2*) of 1.5 p.e., the L1 trigger rate would be 13.9 kHz per LCM and the total trigger rate would be 51 kHz. Each OM would generate 1.1 Mb/s. The corresponding readout fraction would be 20% i.e. a factor 5 reduction in data volume. Table 4.6 details the expected trigger and data rates, under these conditions. Also included are rates for other choices of the high threshold *th2*; in particular, setting the high threshold to 2 p.e. would yield a data reduction factor of 30.

High threshold	1.0 p.e.	1.5 p.e.	2.0 p.e.
L1 trigger	102 kHz	14 kHz	2.5 kHz
L2_2storey	481 kHz	9 kHz	281 Hz
L2 trigger	649 kHz	37 kHz	4 kHz
total trigger	750 kHz	51 kHz	6.5 kHz
data rate per OM	5.1 Mb/s	1.1 Mb/s	0.2 Mb/s
data rate per LCM	15.2 Mb/s	3.4 Mb/s	0.6 Mb/s
data rate per MLCM	76 Mb/s	17 Mb/s	2.7 Mb/s
data rate per string	457 Mb/s	102 Mb/s	16.4 Mb/s
total data rate	4.6 Gb/s	1 Gb/s	164 Mb/s
readout fraction	96 %	20 %	3 %

Table 4.6 : Expected trigger and data rates, based on the test 1.6 results, as a function of the higher threshold (*th2*). The trigger rates are the rates seen by any OM: the LCM trigger rate for L1, and the full detector rate for L2. The cutoff for the trigger was set to 500 kHz.

Given the limited amount of data available on the background rates, especially their long term seasonal behaviour, significant variations in the background rate cannot be excluded. For example, if the overall singles rate were to increase by a factor two, the total trigger rate would increase from 51 kHz to 137 kHz. The readout fraction would be 40% and the data rate would be 67 Mb/s per MLCM. In such a situation it may be desirable to tighten the trigger further; for example, if the higher threshold (*th2*) is increased from 1.5 p.e. to 2.0 p.e., the total trigger rate is reduced from 137 kHz to 17 kHz, and the fraction of the data stream selected is reduced from 40% to 6%. The corresponding data rate is reduced from 67 Mb/s to 11 Mb/s per MLCM. Such an example illustrates the importance of a flexible trigger which can be adjusted to cope with changes in the background baseline rate.

Deadtimes and data losses

The rates in table 4.6 take account of (small) losses due to deadtimes in the system. Deadtime occurs:

- in the front-end chips due to:
  - the time to store pulses in the ARS buffers (170 ns),
  - full SPE buffers (waveform events are treated as SPE events if both waveform buffers are full) and,
  - the output limit of the ARS.

- The deadtime due to the input data rate is less than 1% with two ARSs per OM for singles rates below 600 kHz.

With the higher trigger threshold at 1.5 p.e., the fraction of waveform events which are converted to SPE events because both of the waveform buffers are full, is 1% for OMs with singles rates below 100 kHz, increasing to 15% at 200 kHz and 65% at 500 kHz. The loss of SPE events due to SPE buffers overflowing while the ARSs digitise the waveform signal is negligible for singles rates up to 300 kHz. Above 300 kHz, more stringent trigger conditions would be needed to prevent loss of SPE data. One possible strategy would be to turn off the waveform mode for OMs with high singles rates, and digitise all of the hits in the SPE mode. This requires reconfiguration of the corresponding ARS (parameter All_spe in table 4.2). Even so, the SPE deadtime remains high for singles rates above 1 MHz because of the ARS output limit (800 kHz of SPE for two ARSs).

Trigger efficiency

The effect of the trigger on the standard ANTARES algorithm for reconstructing muons with energies of 200 - 300 GeV intersecting the inner detector volume has been studied. As the muons which are selected by the algorithm have typically ten signal hits with average pulse heights around 3 p.e., half of them with at least 1.5 p.e., a trigger requiring one or two large pulses is very efficient. Indeed, the asymmetric pair trigger has been shown by simulations to be highly efficient (90-97% depending on *th2*) for reconstructed signal events with zenith angles smaller than 80 degrees. The trigger between adjacent storeys is somewhat less efficient (71-94%), and nearly all events selected by that trigger would also be selected by the asymmetric pair trigger. Combining the two L2 triggers gives the relative efficiencies shown in table 4.7 (91-99%).

High threshold	0.3 p.e.	1.0 p.e.	1.5 p.e.	2.0 p.e.
relative trigger efficiency	99 %	98 %	96 %	91 %

Table 4.7 : Relative trigger efficiencies for reconstructed muons with energies of 200 - 300 GeV intersecting the inner detector volume as a function of the higher threshold (*th2*).

Clock system

The main purpose of the clock system is to provide a common clock signal to all ARSs. In addition, it can distribute a (small) number of other signals to the LCMs. These signals can be addressed to a single LCM or all LCMs simultaneously. They are used to synchronise the ARSs, to assign (on-shore) the GPS time to the data, to calibrate the (slave) clocks in the LCMs and to control the off-shore data flow.

At regular time intervals (at least every 800 ms), the internal clock registers of the ARSs are reset using a common RTS signal. The number of these signals is maintained locally by a counter in each LCM. The values of these counters are added to the data. The counters are reset at the start of each data taking run using a clock signal (SOR) which is addressed to all LCMs. The GPS times of the SOR and the subsequent RTS signals are determined with an accuracy of 1 µs. The values of the RTS counters and the GPS times of the RTS signals are used off-line to assign the GPS time to the data.

The (relative) time offset of the slave clocks are determined by measuring the propagation delays of a calibration signal. This signal is distributed through the clock system and returned by one of the slave clocks. The slave clock is preselected using another clock signal which is addressed to the corresponding LCM. The delay is measured on-shore. The detailed specifications for the clock calibration can be found in ref. [10].

The off-shore data flow is controlled by a disable (enable) signal which is addressed to all LCMs and hence all ARSs in the readout system. These signals are used in combination with the *start* and *pause* (*stop*) commands. They are also used in coincidence with the clock signal which triggers an optical beacon (OB). This signal is addressed to the LCM responsible for the OB; it does not require (re)configuration of the OBs. The ARS enable signal is offset between - 5 µs and + 5 µs relative to the OB trigger, and the ARS disable signal is offset between 0 and 10 µs.

The clock system is also used to access the off-shore DAQ hardware independent of the Ethernet connection, e.g. to force a (local) reboot. The GPS time is distributed to various processors in the on-shore data processing and run control systems through the PCI bus of its processor (not shown in fig.4.1) [9]. The accuracy of the absolute time(s) is better than 1 µs. The (slave) clocks in the LCMs are configured at *power on*.

Instruments

The instruments which are used for the monitoring of the detector are explained in detail in chapter 6. In general, the instruments are controlled using the universal board (**UNIV1**). This board has a micro processor on which the **microSC** process runs. The **UNIV1** circuitry is also mounted on other hardware devices in the system (see chapter 5 on Electronics) and these devices can also be regarded as instruments. The interface between the **microSC** and the **daqLCM** processes proceeds through the LCM backplane using the RS485 link and the MODBUS protocol. The frequency of the measurements is typically once per minute; it is defined by the operator and stored in the general database. The instrumentation for the optical beacon consists of two parts: the first part, based on the UNIV1 circuitry, is used to control the setting of the LED driver. The second part is an ARS mother board used to read out the (fast) PMT inside the optical beacon. A list of the instruments with their readout parameters and the corresponding data sizes is given in table 4.8. It is not foreseen to read out individual parameters.

Instrument	Parameters	total size (Bytes)
COMPASS_MB	cap, pitch, roll, Bx, By, Bz, temp, error, checksum	23-40
	humidity	2
	Temp1, Temp2	4
	T_ARS_MB1, T_ARS_MB2, T_ARS_MB3, (T_ARS_MB4)	6 (8)
	HV_PMT1, HV_PMT2, HV_PMT3	6
LCM_DWDM	temp, Ibias, Imodu, I_TEC, V_TEC, power, Tloc, V1, V2	??
SCM_DWDM	temp, Ibias, Imodu, I_TEC, V_TEC, power, Tloc, V1, V2	??
LCM_ACOUSTIC (1-3)	Elementary cycle, time and amplitude, positioning cycle	7 + n x 12
SCM_ACOUSTIC (1-6)	Elementary cycle, time and amplitude, positioning cycle	7 + n x 12
ACOUSTIC_SVEL-CTD	velocity, conductivity, temp and pressure	27
ACOUSTIC_PRESS	pressure	7
ACOUSTIC_SVEL	velocity	8
BEACON_ARS_MB	SPE event	6
	WF event	263
LCM_POWER_BOX	V48, V48s, V12, V5.5, V5, V3.3, V2.5, V1.8 I48, I48s, I12, I5.5, I5, I3.3, I2.5, I1.8	??
SCM_POWER_BOX	V48, V48s, V12, V5.5, V5, V3.3, V2.5, V1.8 I48, I48s, I12, I5.5, I5, I3.3, I2.5, I1.8	??

SPM	Vin, V1out, V2out, V3out, V4out, V5out, V6out, I1out, I2out, I3out, I4out, I5out, I6out, T1, T2, T3, IACleak, IDCleak	18
	status	3

Table 4.8 : Readout parameters of the instruments ('**n**' indicates a number of repetitions).

The instruments which need to be configured are listed in table 4.9 with their configuration parameters and the corresponding data sizes. For the ARS motherboard of the OB, only a summary of the ARS configuration parameters is given.

Instrument	Parameters	total size (Bytes)
COMPASS_MB	HV_PMT1, HV_PMT2, HV_PMT3	6
LCM_ACOUSTIC (1-3)	Nel, F_Rx, autoGain, anaGain, digGain, th, maxDD, start, stop	2 + n x 27
SCM_ACOUSTIC (1-6)	Nel, DSP, F_Rx, autoGain, anaGain, digGain, th, maxDD, start, stop F_Tx, level, delay, duration	$3 + n_1 \times 37 + n_2 \times 12$
BEACON_MOTHERBOARD	intensity	??
BEACON_ARS_MB	Trig0_th, PSD_th, Spe_clk, Sel_pled_clk, En_acq, All_spe, All_wave, PSD_tot, Burst, Sel_pulse, En_pulser, All_t1b	239/8
SPM	status	1
	reset	0

Table 4.9: Configuration parameters of the instruments ('**n**' indicates a number of repetitions).

DAQ hardware

The network architecture of the off-shore DAQ system has a star topology, consisting of a 1--1 optical connection in the JB for each detector string, an optical (de)multiplexer in each SCM, an electronic data router in each MLCM (switch) and a processor in each LCM. The data flow in the detector is shown in figure 4.5.

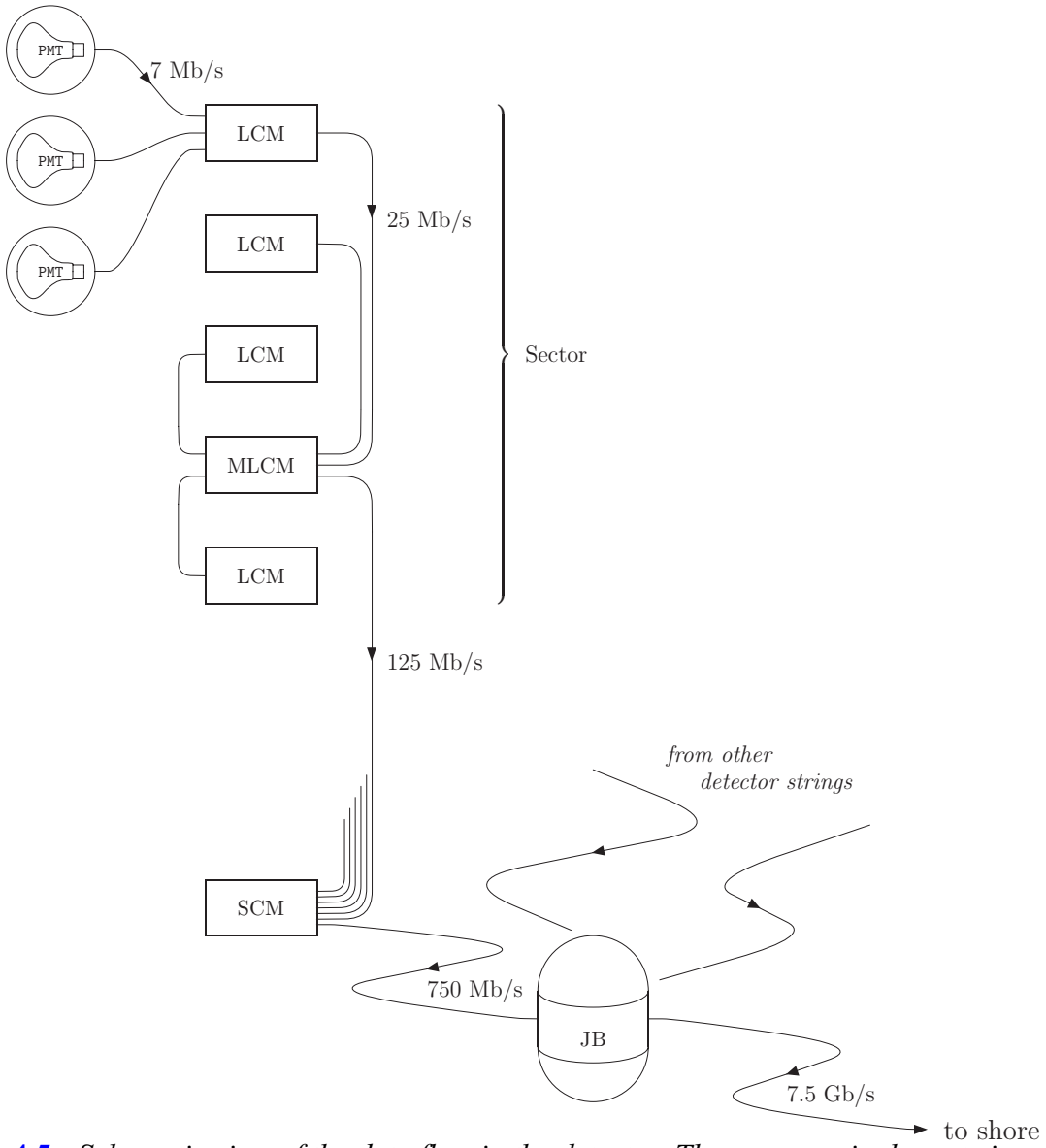

Figure 4.5 : *Schematic view of the data flow in the detector. The average singles rate is assumed to be 70 kHz and the fraction of WF hits 2%. The detector consists of ten strings.*

The network architecture on-shore consists of an optical (de)multiplexer for each detector string, an electronic data router (switch), a data processing farm and a data storage facility. The data flow in the on-shore data processing system is shown in figure 4.6.

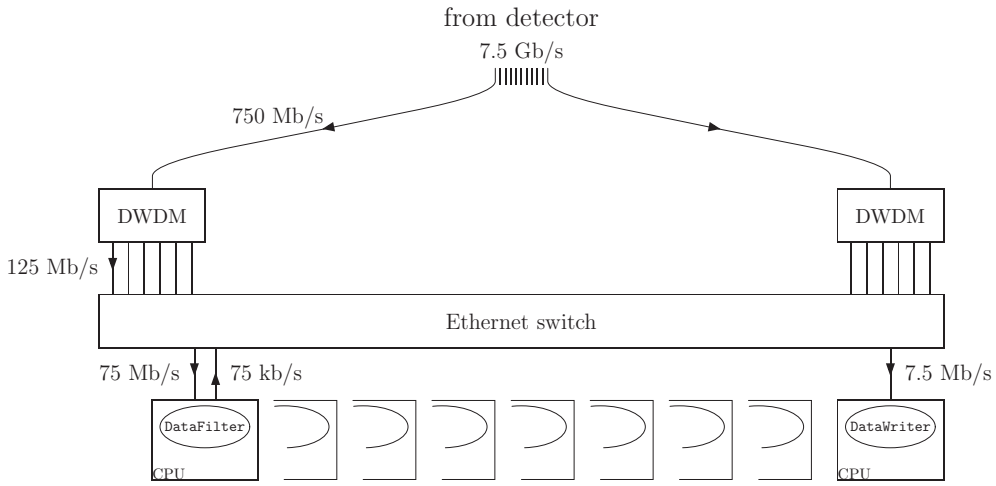

Figure 4.6: *Schematic view of the data flow in the on-shore data processing system.*

The Internet protocol (IP) is used for the communication between the processors. TCP will be used for (command) messages. For the transfer of the ARS data, UDP (User Diagram Protocol) could be used instead. UDP requires less overhead than TCP but is less reliable. For instance, it does not require a connection (socket) between the **DataFilter** CPU and each LCM processor. A higher (effective) bandwidth can be obtained at the cost of some packet loss. This loss should be less than 2%.

Between the detector and the shore station, the **1000Base-LH** standard is applied where 7 channels (6 MLCMs and 1 SCM) are multiplexed into one optical fibre using the dense wavelength division multiplexing (DWDM) technique. For the on-shore local area network (LAN), the **1000Base-SX** standard is applied. Off-shore, the Motorola **MPC8xx** processors are used with the **VxWorks** operating system. On shore, standard PCs are used with the **Linux** operating system. The conversion between **BigEndian** (Motorola) and **LittleEndian** (Intel) will be done by the processors on shore.

LCM processor

The LCM processor (LCM_DAQ/SC in chapter 5) is the hardware interface between the ARSs and the online data processing system. It hosts the **daqLCM** process. A schematic view of the LCM processor is shown in figure 4.7. The LCM processor has an Ethernet port for the connection to shore which is compatible with the Internet protocol. Inside the LCM, a serial port is used to transport the slow control signals. The connection to the ARSs, the clock, the trigger and other instruments is made through the LCM backplane. A detailed description of the LCM backplane is given in chapter 5. The specific hardware for the readout of the ARSs is implemented in a high density Field Programmable Gate Array (FPGA). The data are temporarily stored in a high capacity memory (SDRAM) allowing the de-randomisation of the data flow. The data are read out from this memory by the **daqLCM** process. From then on, the **daqLCM** process schedules the data flow. A Real Time Operating System (RTOS) is used to programme a fixed data transfer rate and schedule slow control actions. The EEPROM contains the data for programming the FPGA, and the flash memory contains the operating system for the processor.

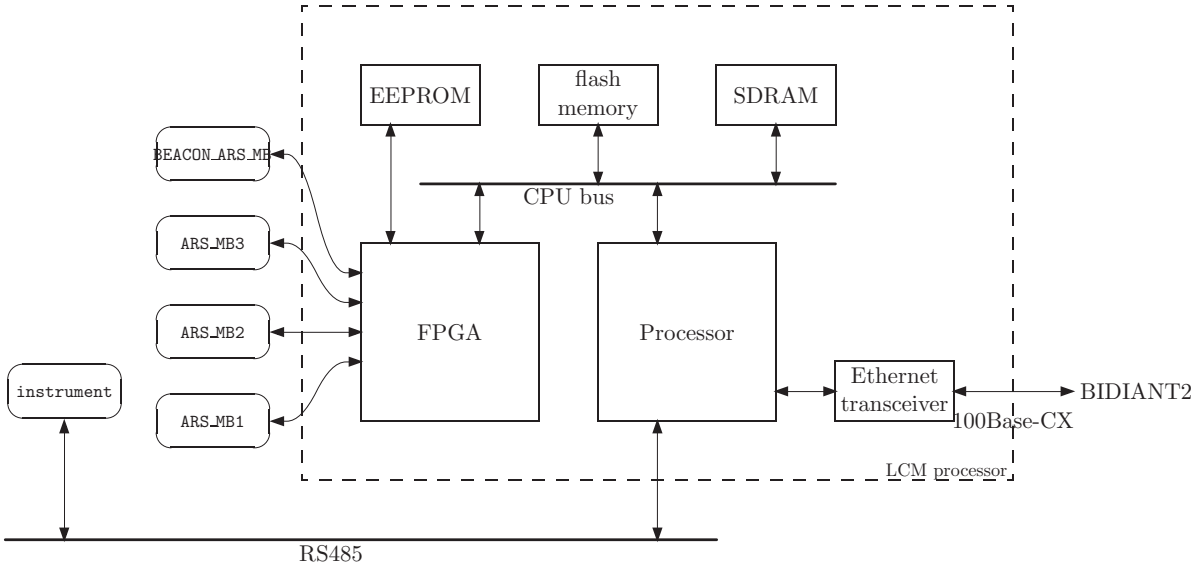

Figure 4.7: *Schematic view of the LCM processor board.*

The present choice for the FPGA is the Altera APEX20K200E. For the processor, the present choice is the Motorola MPC860P in combination with the VxWorks operating system. The MPC860P is a low power (650mW at 50 MHz) RISC processor featuring a communications processor module and a Fast Ethernet Controller (FEC). Together with a boot flash memory and the SDRAM, it constitutes a network-ready system. It has an on-chip100Mb/s Ethernet controller, up to four Serial Communication Controllers (SCC) and a Serial Peripheral Interface (SPI) controller. The Ethernet port is used for the connection to the MLCM switch. The slow control actions can be carried out using one of the SCC or SPI ports. The MODBUS protocol is obtained by connecting an RS485 driver chip to one of the SCC ports and implementing the corresponding software. Figure 4.8 shows a block diagram of the ARS readout (RARS) logic with the connections to the processor and the SDRAM. The data arrive from the serial outputs of the ARS chips asynchronously to the RARS, where they are buffered in the memory of the FPGA through 6 Link Controllers, one for each ARS. The Link Controller checks the data consistency and checks that enough memory space is available, then it writes the data to memory and updates its memory status flag. If inconsistent data are detected, an error is reported in the Control & Status Register. In normal mode, the inconsistent data are discarded, whereas in debug mode they are kept for off-line analysis. The Link Controller will re-synchronise on the next consistent data set (hit).

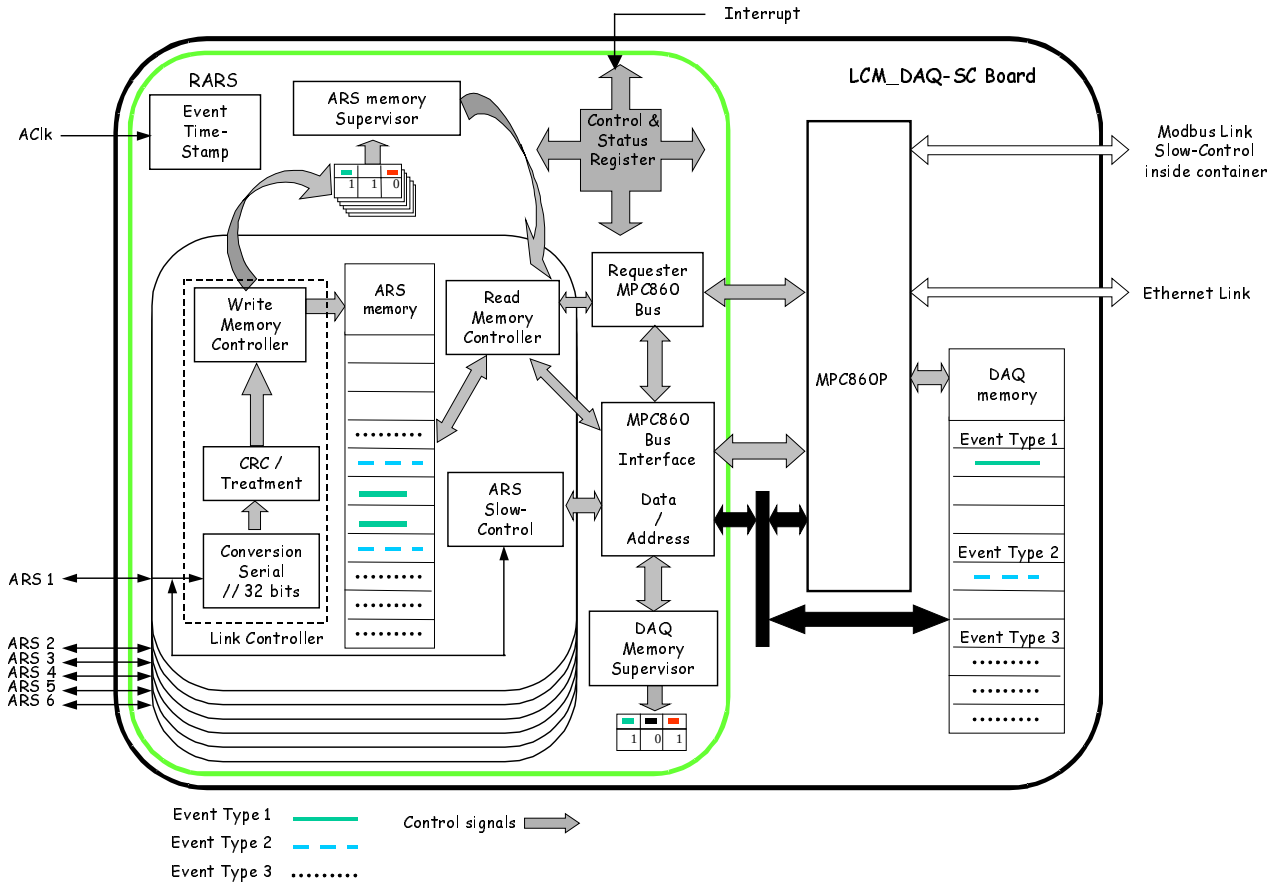

Figure 4.8: *Block diagram of the readout logic.*

The DAQ memory is implemented in the SDRAM. It is organised in different areas corresponding to the hit type and the ARS identifier. Each area acts as a FIFO which is written by the RARS device and read by the MPC860 internal FEC device. A memory of 64Mb allows the storage of 4 seconds of data at the maximal ARS output rate. If a FIFO becomes full, it is reported to the DAQ Memory Supervisor. Additional incoming data corresponding to this memory area are then lost. At initialisation, the memory space is reserved by the Operating System, and the corresponding pointers are written inside the Read Memory Controllers. The ARS Memory Supervisor monitors the filling of the FPGA memory for each ARS. It decides which channel should transfer its data to the DAQ memory. The corresponding Read Memory Controller is triggered which in turn requests control of the CPU bus via the MPC860 Bus Requester. When the data transfer is completed, the memory in the FPGA is cleared and the DAQ memory pointer is incremented (modulo the reserved space) and saved for the next transfer. The MPC860 Bus Interface performs the actual transfer of the data to the DAQ memory. It uses a processor-specific protocol; data transfers are performed using either the ``burst mode'' in which data are transferred in bursts of four 32-bit words, or the less efficient ``single mode'' in which data are transferred as single 32-bit words. The MPC860 Bus Interface can operate either in master or slave mode. The master mode is used to transfer the ARS data from the FPGA to the DAQ memory, while the slave mode is used to initialise and read the status of the RARS device. Whenever an error is detected or a specific external signal is activated, the processor may undergo a hardware interrupt. It is the responsibility of the interrupt service routine to read the RARS status and detect the interrupt source reported in the Control & Status Register. System integrity (e.g. SDRAM refresh) is ensured by the internal bus arbiter of the processor. The master priority is configured at initialisation. The MPC860 Bus Requester should relinquish the bus in the clock cycle following the processor request. The ARS Link Controllers are re-synchronised with the system clock inside the serial to parallel converter. From then on, all internal operations are synchronous with respect to the system clock. The CPU bus is also synchronous with the system clock.

For development and diagnostics, on-board interfaces provide direct access to the CPU board resources (not shown in figure 4.7). They cannot be accessed when the board is plugged into the LCM. The background debug mode provides access to the development support functions which are implemented in the processor hardware. For instance, it gives access to devices connected to the CPU bus (e.g. flash memory, SDRAM) and all internal registers. It is used to debug the prototype and to test production boards. The process monitor can be used to redirect all print commands issued by the OS drivers at boot time or during normal operation to the user's terminal. Depending on the boot programme, it is also possible to enter new boot parameters interactively. The FPGA programming file can be loaded directly from a PC using a JTAG port. The EEPROM can be programmed from a PC via another JTAG port.

The RARS is also used for the slow control of the ARSs. The associated registers are read or written in slave mode. In order to assign the GPS time to the data off-line, a counter of the reset time stamp signal is used to provide a coarse-grained time stamp with a dynamic range of 10 hours. This counter is reset through the clock system at the start of each data taking run. The clock signal is used to dimension the data arrays of SPE and WF hits according to a predefined time frame. The corresponding IP address is assigned to each array by the **daqLCM** process. In order to relate off-line the off-shore trigger signal to the ARS data, the L1, L2 and RoR signals are time stamped using the same clock signal. The present version of the operating system does not allow multiple processes to run on a single CPU. Instead, it provides multiple threads for a single process. This requires the slow control and DAQ functions to be integrated in a single process **daqLCM** (see section DAQ software). It is possible to make a telnet connection to the processor. This allows system tests to be performed or new software to be downloaded. At *power on*, the data for programming the FPGA are downloaded from the EEPROM and the operating system is downloaded from the flash memory. As a result, both the FPGA and the processor are configured. It is not yet specified what additional software is loaded from the flash memory. The procedure for downloading software at *configure* has not yet been defined.

MLCM switch

The MLCM switch is shown schematically in figure 4.9. Its main function is the merging of all Ethernet links (100 Mb/s) coming from the LCM processors of a sector into a single Ethernet link (1Gb/s) to shore. The links to the LCM processors are made through the optical fibres in the string cable using fast Ethernet transceivers. The transceivers inside the MLCM are mounted on a single board (LCM_BIDICON). This board and the processor inside the MLCM are connected to the MLCM switch using PECL signals over the LCM backplane. The connection to the 1000Base-LH Ethernet MLCM transceiver (MLCM_DWDM) is made by a direct electrical bridge between the two corresponding cards. The present choice for the switch is a combination of the Allayer AL121 (eight 100 Mb/s ports) and the Allayer AL1022 (two 1Gb/s ports). The necessary buffer capacity has not yet been specified. The switch is configured at *power on*.

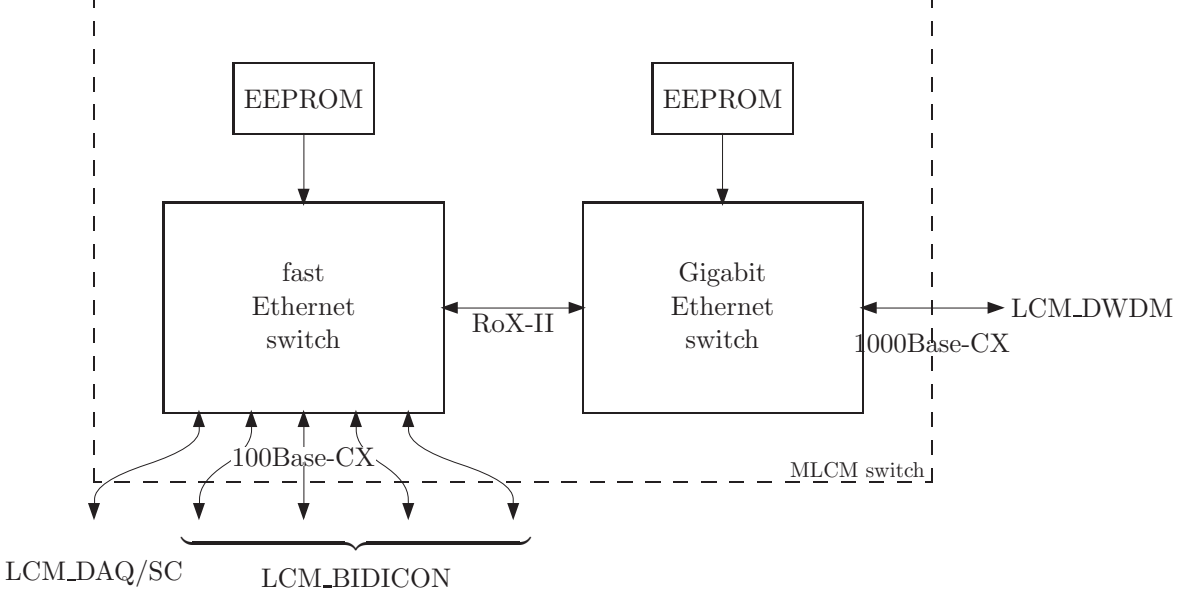

Figure 4.9: *Schematic view of the MLCM switch.*

MLCM transceiver

The main function of the MLCM transceiver (MLCM_DWDM described in Chapter 5) is to establish a 1 Gb/s Ethernet link to shore. The optical receiver consists of a analogue pin diode (APD) with a built-in trans impedance amplifier (TIA). The output of the TIA is AC-coupled via a single stage low-pass LC filter to a limiting amplifier. PECL signals are used for the transmission of the amplifier output to the MLCM switch. The optical transmitter consists of an internally cooled distributed-feedback (DFB) laser. The laser driver contains a bias circuit and a modulation circuit. In order to minimise the number of connections, the slow control circuitry is mounted on the board. A unique pair of wavelengths is assigned to each MLCM in the string, such that the (de)multiplexing in the SCM can be achieved optically. In order to maintain uni-directionality of the optical data transmission, two fibres are needed between the MLCM and the SCM, between the SCM and the JB, and between the JB and the shore. The MLCM transceiver is configured at *power on.*

Dense wavelength division multiplexing

The dense wavelength division multiplexing (DWDM) technique is used to (de)multiplex the 7 Ethernet channels (6 MLCM plus 1 SCM) between each detector string and the shore into one optical fibre pair. One 8 x 2 DWDM module (SCM_DWDM_MUX/DEMUX in chapter 5) is located in each SCM. Identical DWDM modules are used in the front-end of the data processing system on shore. They are organised in a crate which has a backplane and a processor. The slow control signals (read-only) are transmitted using a RS485 link and the MODBUS protocol. The results are written to the **readings table** of the general database via the **DBwriter** interface. The connection in the JB between the main electro-optical cable and the interconnection cables is 1-to-1 and does not require (additional) optical components.

On-shore switch yard

The main purpose of the on-shore switch yard is to route the data sent by the **daqLCM** processes to the **DataFilter** processes. For simplicity, it is not shown in fig. 4.1. It has a total of about 200 physical channels: 70 channels corresponding to the number of MLCMs and SCMs in the (10-string) detector 100 channels for the processors running the **DataFilter** programme and a few additional channels for the processors running the **DataWriter**, **RunControl**, etc. programmes. It relies on IP addressing of the Ethernet protocol (level 2). This requires that the data acquisition

system in the LCM determine the destination addresses of the data as a function of their time stamp. It should handle a continuous data rate of about 10Gb/s. The necessary buffer capacity is not yet specified.

Data processing farm

The data processing farm consists of about hundred PCs, each PC running the **DataFilter** programme. The expected input data rate per PC is estimated at 100 Mb/s; the expected output data rate is 100 kb/s, corresponding to a reduction factor of about 10^3. With TCP/IP, a total of 310 Ethernet channels is maintained by the **DataFilter** process, one for each LCM (SCM) processor. The performance of the processor and the **DataFilter** programme are critical. The IO rate should be at least 300 Mb/s and the processor speed at least 700 MHz. In order to minimise dead time due to activities of other processes or daemons, these processors should run a restricted version of Linux. A separate PC running a **Dispatcher** programme is used to collect the output data of the **DataFilter** processes and distribute these data to various clients. Amongst these clients, the DataWriter process takes care of saving the data on disk. Optionally, a **Dispatcher** programme can be run on every processor to collect the input data and store them in (shared) memory. This allows a continuous 10 -100 s history of the data to be kept in memory so that these data can be stored on the local disks in case of a (delayed) external trigger signal. With 256 MB of RAM, **20** seconds of real data can be kept continuously in memory. In order to store 20 minutes of data, the disk space of every processor should be 32 GB or more and the speed of writing to disk should be at least 100Mb/s.

Data storage

The disk space foreseen for data storage is 1 TB. This corresponds to more than one week of data taking (at 7.5 Mb/s). For permanent storage (and distribution), the data written to disk will be copied to tape. A DLT robot with eight 45 GB tapes can store the data taken during several days. It is planned to have a high bandwidth (100 Mb/s) data link to the outside world. Through this link, the output of the **DataWriter** can be sent to any European computer centre in real time, providing rapid worldwide access to the data.

DAQ software

The data acquisition and processing software is written in C or C++. It incorporates the DAQ model introduced in section DAQ model using a language system for concurrent hierarchical finite state machines [1]. The different procedures necessary for a transition between two states are implemented in these programmes. The higher level protocol for the communication between processes has not yet been chosen (**MATRA**[4] or **Dispatcher** [5]). In the following, the **Dispatcher** protocol is assumed. The list of the programmes is given in table 4.10. A more detailed description of these programmes is given in the following paragraphs.

Programme	Description	Input	Output
Dispatcher	Data distribution	any	any
daqLCM	Offshore DAQ and Slow Control	ARS trigger clock instrument *command*	{*hit*}[1] {*value*} *message*
daqSCM	Offshore slow control	instrument clock *command*	{*value*} *message*
microSC	Hardware device control	device *value*	*value* device
daq3D	Acoustics data acquisition	hydrophone	*distance*
DataFilter	Physics data filter	{*hit*} *command*	{*event*} *message*
ProcMan	Process manager	*command*	*message*
DataWriter	Write event data to disk	*event* *command*	disk *message*
DBwriter	Write entries in database	{*value*} *command*	database *message*
clockIO	Clock system interface	clock *command*	*message*

[1] The {} refer to an array of items.

Table 4.10 : DAQ software.

Dispatcher

The **Dispatcher** is used as a general server programme to collect and distribute data from (to) client processes [5]. It uses TCP/IP and runs as a stand alone process. It should be started during the boot up of the processor. It can be considered as an extension of the operating system.

daqLCM

The **daqLCM** is the process running on the processor in the LCM. Its main task is to handle the readout of the ARS chips, the organisation of the data in 10 -20 ms time frames and the transmission of the data to the corresponding on-shore processors. This process also takes care of the local slow control actions, e.g. the reading of instruments. The results are written to the **readings table** of the general database via the **DBwriter** interface. It runs in event driven mode, where during normal operation the events initiate the transfer of ARS data. A limited number of slow control actions can be scheduled. It still reacts to simple commands from the **RunControl** process in order to *pause*, *stop*, *start* or *continue* the data taking. The process receives its running parameters through the *configure* message with additional data. The configuration of the PMTs, ARSs, the off-shore trigger and all instruments is also handled by the **daqLCM** process. The data sent to shore are contained in a buffer; each buffer has a generic header consisting of a specifier for the total length of the buffer and an object identifier.

length	U32	Total size of this buffer
type	U32	Object identifier
body		Encoded data

U= unsigned integer

The ARS data ({*hit*} in fig.4.1) are formatted by the **daqLCM** process as follows:

SPE	status	U8	ARS header word	WF
	time stamp	U24	ARS raw time stamp	
	TVC	U8	ARS raw TVC	
	charge	U8	ARS raw amplitude	
	N	U8	Number of WF samples	
	body	U16	WF samples	

The SPE and WF data are sent in separate buffers. The clock signal is used to dimension the data arrays of SPE and WF hits according to a predefined time frame and to assign the corresponding IP address. A copy of the SPE data contained in the WF data block is added to the SPE buffer. Both buffers have an extended header in which the type parameter specifies the type of data (ARS_SPE or ARS_WF) and the container parameter specifies the origin. The container parameter is specified in the address table of the data base. In order to assign (off-line) the GPS time to the data, the header of each buffer also contains the value of the counter of the ARS reset time stamp (RTS) since the start of the run.

length	U32	Total size of this buffer
type	U32	ARS_SPE or ARS_WF
RUN	U32	Run number
container	U32	Container identifier
ARS_MB	U32	ARS motherboard
RTS counter	U32	RTS counter
N1	U32	Number of digitised hits
N2	U32	Number of hits in this buffer
status	U32	Status of this buffer
body		ARS data (if any)

The sizes of these buffers are determined by the predefined time window and the singles rate.

With a singles rate of 70 kHz, the SPE buffer size is 4 - 8 kB or a 10 - 20 ms time window. With a 2 % fraction of WFs, the size of the WF buffer varies in the range 4 - 8 kB. These sizes yield an efficient use of the TCP/IP protocol. In order to relate the off-shore trigger signal to the ARS data off-line, the time stamped L1, L2 and RoR signals are sent together with the ARS data. For the PMT calibration, each ARS is configured to generate a single pulse synchronous with the reset time stamp signal. This pulse is used to drive the LED that illuminates the photocathode of the PMT. As a consequence, data with a time stamp value around the PMT transit time (100 ns) can be directly identified as calibration data (the probability of a background hit is less than 1 % in a 100 ns window). These data can be collected locally by the **daqLCM** process, averaged for each PMT and stored in the readings table of the general database (PMT) via the DBwriter process. As a result, the PMT calibration can be considered as instrument data. The instrument data ({*value*} in figure 4.1) are also collected by the **daqLCM** process. The current value of the RTS counter is added to these data. This allows the instrument data to be correlated with the physics data. The instrument data are formatted as follows.

size	U32	Size of this block
device	U32	Device identifier
parameter	U32	Parameter identifier
RTS	U32	RTS counter
status	U32	Status of this block
body		data (if any)

The corresponding data flow is small compared to that of the ARS data. The buffers are sent to shore by the **daqLCM** processes with a frequency between 1/s and 1/min. The buffers have an extended header.

length	U32	Total size of this buffer
type	U32	SC
RUN	U32	Run number
container	U32	Container identifier
status	U32	Status of this buffer
body		data (if any)

The **daqLCM** process is started at boot up of the processor or by a *rsh* from the **RunControl** programme.

daqSCM

The **daqSCM** process is similar to the **daqLCM** process except that no readout of ARSs is foreseen. It does read out the various instruments inside and outside the SCM container. In addition, it controls the String Power Module (SPM). The default (hardware) settings can be by the **daqSCM** process the timeout event (see section DAQ model).

microSC

The **microSC** process runs autonomously on the microprocessor of the **UNIV1** board. It allows parameter values to be read from or written into the hardware devices. The **microSC** process is started at *power on*.

daq3D

The **daq3D** process handles the data acquisition of the acoustic system (see chapter 6). It runs on

a separate processor (DSP). It configures the acoustic signal emitter and receiver and controls the emitting and receiving of the acoustic pulses. It can be considered as an instrument where the measured parameter value corresponds to a distance. The interface to the **daqLCM** process is not yet defined. The **daq3D** process is started at *power on*.

DataFilter

The main purpose of the **DataFilter** process is to detect the physics events in the data and % remove the uninteresting background data. Each process receives the data from all the daqLCM processes corresponding to a common time frame. A reduction factor of 10^{-3} (10^{-4}) yields 30(3) TB of (filtered) data per year. The data containing the physics events are sent to the central **DataWriter** process. **DataFilter** runs in event driven mode, where during normal operation the events initiate the processing of ARS data. It still reacts to simple commands from the **RunControl** process in order to *pause*, *stop*, *start* or *continue* the data processing. The process receives its operating parameters through the configure command with additional data. When processing calibration data, the filtering algorithm can be bypassed. The performance of the **DataFilter** programme can be greatly enhanced if the data are time ordered. In order to reduce the data output, only time correlated data are sent to the **DataWriter** process. The **DataFilter** process is started by the **ProcMan** process.

ProcMan

The **ProcMan** programme is the process manager for the **DataFilter** processes. It facilitates the launching and aborting of the **DataFilter** processes. It is started at *power on* of the processor or with a remote shell from the central **RunControl** programme. It is command driven and accepts commands from the **RunControl** programme.

DataWriter

The **DataWriter** process collects the filtered data from all **DataFilter** processes and writes them to disk. It runs in data driven mode but it still reacts to simple commands from the **RunControl** process in order to *pause, stop, start* or *continue* the data processing. The *start* message has an additional data word specifying the new run number. The *stop* message causes the **DataWriter** process to close its current output file. The **DataFilter** process is started by a *rsh* from the **RunControl** programme.

DBwriter

The **DBwriter** process transfers the measured parameter values to the database system. The measured values are sent from the **daqLCM** processes and the **clockIO** process to a (central) **Dispatcher** which transfers the data to the **DBwriter** process. The data contain the run number, the corresponding value of the RTS counter, the container, device and parameter identifiers, and the measured value(s). It determines the table name from the data (readings, acoustic or clock) and makes a SQL request to insert the new measurements into the database. The **DBwriter** process is started by a *rsh* from the **RunControl** programme.

clockIO

The **clockIO** process is the interface between the run control system and the central clock system (see section Clock system). It is a client of the **Dispatcher** running on the host of the **RunControl** process. It handles the corresponding commands generated by the operator and sends back a message containing the GPS time of the transmission of the clock signal. The GPS times of the SOR signal and the subsequent RTS signals are stored via the **DBwriter** in the **operator** table of the

general database. The number of RTS signals since the start of a data taking run is counted by the **clockIO** process and added to the database entry. The calibration of the (slave) clocks in the LCMs is controlled by the **clockIO** process independently of the data acquisition. The measured delays are collected by the **clockIO** process, averaged for each LCM and stored in the **clock** table of the general database via the **DBwriter** process. Through the **clockIO** interface (and the clock distribution system), the off-shore data flow is controlled using the enable (disable) signals of the ARSs. The **clockIO** process is started by a *rsh* from the **RunControl** programme.

Control system

The main purpose of the control system is to initiate and verify the configuration of the detector, the initialisation of all processes and the changes between the possible states of the DAQ system. The front-end of the control of the data acquisition system and the detector proceeds through Graphical User Interfaces (mostly) written in **Java**. The main requirement of these interfaces is their ability to be operated remotely. The list of the GUIs is given in table 4.11. A more detailed description of these GUIs is given in the following paragraphs.

GUI	Description	Input	Output
RunControl	General run control	user database *message*	screen database *command*
Logger	Message saving	*message* user	screen disk
SettingsEdit	Edit the settings parameters	user database	screen database
MonitorEdit	Edit the monitoring parameters	user database	screen database
TriggerEdit	Edit trigger parameters	user database	screen database
FilterEdit	Edit filter parameters	user database	screen database
ExpertControl	Expert control of the detector	user *message* *value* database	screen *command*
Monitor	online monitor control	user rsh	*command* rsh
Display	Histogram display	user disk	screen
Logbook	Electronic logbook	user database	screen database

Table 4.11: Graphical user interfaces.

RunControl

The **RunControl** is the main GUI to the data acquisition system and the detector. It features multi-user operation with the restriction that only one user can be master. Only the master is allowed to modify the actual state of the data acquisition. All other users are only allowed to monitor the running conditions. It implements the DAQ model of section DAQ model. It can generate the *on, off, configure, start, stop, pause* or *continue* events shown in fig. 4.2. For every event, an entry is written in the **runlist** table of the general database containing the GPS time. Prior to the *configure* command, the parameters in the detector configuration tables can be selected. The *configure* event causes the **RunControl** programme to extract the configuration parameters from

the database and to distribute them to the corresponding clients. The *start* command has an additional data word specifying the new run number. At each *start* event, an entry is written in the **runoptions** table of the database specifying the pre-selected parameters of the run. The updating of the run number is controlled by the **RunControl** programme. In order to archive data efficiently, the run number is updated automatically every 4 hours. The **DataWriter** process is then instructed to close its current output file and open a new one. The size of the corresponding disk-files is about 20 GB. In the off-line analysis, these periods and the corresponding disk-files can be identified by their run numbers.

Logger

The **Logger** process collects the messages from all the processes. It displays the messages on a screen and/or saves them on disk.

SettingsEdit

The **SettingsEdit** is the GUI for the detector settings. The values in the parameter column of the **settings** table of the general database can be retrieved, modified and (optionally) stored as a new entry. Normally, it is started from the **RunControl** (when in state **OFF** or **STANDBY**). It can run in standalone mode if the **RunControl** programme is not running.

MonitorEdit

The **MonitorEdit** is the GUI for the detector monitoring. The values in the frequency, min. and max. columns of the **monitor** table of the general database can be retrieved, modified and (optionally) stored as a new entry. Normally, it is started from the **RunControl** (when in state OFF or **STANDBY**). It can run in standalone mode if the **RunControl** programme is not running.

TriggerEdit

The **TriggerEdit** is the GUI for the settings of the off-shore trigger system. These settings can be retrieved from the **trigger** table of the database, modified and (optionally) stored as a new entry in the same table.

FilterEdit

The **FilterEdit** is the GUI for the settings of the data filtering algorithm. These settings can be retrieved from the **filter** table of the database, modified and (optionally) stored as a new entry in the same table.

ExpertControl

The **ExpertControl** is the direct GUI to the detector. The actual settings of a hardware device can be modified and (re)read. It can send a *command* to the **daqLCM** process via the **Dispatcher** running on the **RunControl** processor. Via the same **Dispatcher**, it receives the *value* data following a read *command*. A possible error *message* is also intercepted.

Monitor

The **Monitor** is the GUI to control the online monitoring processes. They can be started (*rsh*) and stopped from this interface. It can request saving of the current histograms on disk.

Display

The **Display** is the GUI to display the histogrammes which are written to disk by the online monitoring processes. It has an option to print its output and provides the facility to compare histogrammes.

LogBook

The **LogBook** is the GUI for entering ASCII text. It archives the entries in the general database and allows their recovery and display.

Online monitoring

The online monitoring system is used to monitor the quality of the data and the stability of the detector. This reduces the time required to detect errors and speeds up the verification procedure after expert intervention(s). The **Dispatcher** running on the **DataWriter** CPU is used to distribute the data to the various monitoring processes. These processes analyse the data and store histograms on disk. They are written in C, C++, Java or other languages. The list of the monitoring programmes is given in table 4.12. A more detailed description of each programme is given in the following paragraphs.

Programme	Description	Input	Output
EqCheck	Data quality check	*event* *command*	disk *message*
FastMon	Monitor physics data	*event* *command*	disk *message*
SlowMon	Monitor parameter values	{*value*} database *command*	disk *message*
a3dOnline	Online event display	*event* user	screen

Table 4.12: Online monitoring software.

EqCheck

The **EqCheck** programme checks the consistency of the data: headers, array lengths, etc. It reports possible errors to the **RunControl** process.

FastMon

The **FastMon** programme analyses the contents of the data and histograms the basic parameters: e.g. singles rates, fraction of waveform events, etc. The results are written to disk. They can be displayed by the **Display** programme.

SlowMon

The **SlowMon** programme analyses the measured parameter values of the hardware devices on the detector strings. It is a client of the **Dispatcher** running on the **RunControl** processor. It histograms the basic parameters: e.g. temperature, low voltages, etc. The results are written to disk. They can be displayed by the **Display** programme. It can act as a watchdog by checking the data values. For this, it needs to obtain the **monitor** parameters from the database. It can send a warning message to the **RunControl** process if necessary. These alarms are signaled to the user of the **RunControl** who in turn can take the appropriate action. It is not clear whether 'hardware' interlocks are needed, i.e. slow control actions without human intervention.

a3dOnline

The **a3dOnline** programme is the ANTARES event display programme linked to the event stream. Its output is displayed on the user's terminal.

Data storage and analysis

The data written to disk by the **DataWriter** process(es) are copied to tape locally. These tapes are used for the detailed off-line analysis of the data. An online analysis of the data is foreseen to obtain the time calibration and the alignment parameters of the detector. For this, the data available on disk are used. This procedure can be repeated off-line using the data stored on tape. The list of the programmes is given in table 4.13. A more detailed description of the programmes is given in the following paragraphs.

Programme	Description	Input	Output
Tapewriter	Copy data to tape	disk database	tape database
DetAlign	Detector alignment	user disk database	database
TimeCalib	Time calibration	user disk database	database
GainCalib	Amplitude calibration	user disk database	database
Recons	Event reconstruction	user disk database	database

Table 4.13: Data storage and analysis software.

TapeWriter

The **TapeWriter** process is a daemon; it monitors the **runlist** table of the general database. When a run is ``closed'', it copies the corresponding disk-file produced by the **DataWriter** process to tape. It prepends the configuration parameters and the events of the corresponding run (cf. **settings**, **trigger**, **filter** and **operator**). For each (un)successfully written tape, it writes an entry in the **tapelist** table of the database.

DetAlign

The purpose of the **DetAlign** programme is to obtain the (relative) positions of all acoustic beacons in the detector. It is a standalone programme which is started and controlled by the user. It analyses the data acquired by the **daq3D** processes. These data are stored in the **acoustic** table of the database. The results are saved in the **alignment** table of the database.

TimeCalib

The purpose of the **TimeCalib** programme is to obtain the (relative) time offsets of all OMs in the detector. It is a standalone programme which is started and controlled by the user. It analyses the data acquired by the **daqLCM** processes during the flashes of the optical beacons. It requires

knowledge of the (relative) positions of all PMTs in the detector. The results are saved in the **calibration** table of the database.

GainCalib

The purpose of the **GainCalib** programme is to obtain the amplitude calibration of all OMs in the detector. It is a standalone programme which is started and controlled by the user. It analyses the (background) hits in the physics (or calibration) data. The results are saved in the **amplitude** table of the database.

Recons

The purpose of the **Recons** programme is to make (online) event reconstruction. It is a standalone programme which is started and controlled by the user. It analyses the output data of the **DataFilter** process. It requires knowledge of the (relative) positions of all PMTs and their calibration.

Database

The **Oracle8*i*** database system is used to organise the detector assembly and integration and to centralise the running conditions of the detector. It is also used to store and retrieve the measured parameter values of the hardware devices during (physics) data taking. In general, the data are stored in two dimensional tables. The database system handles the links between these tables. The keywords for these links are hardware labels, the universal time and the run numbers. The access control is handled by identifying two kinds of users, those with read and write access and those with read access only. The user interfaces are written using the WEBDB tool of the Oracle system. The direct read or write access of the **DBwriter**, the various **GUI**s and the data storage and analysis programmes require a C, C^{++} , and Java pre-compiler/interface to the database system.

Detector assembly and integration

In general, the detector parts are produced, tested and stored in different places. For each detector part, a web interface is used to enter the test data and its history either manually or automatically into the database. This allows to know which components are available, tested and qualified. For the assembly and integration, a list of qualified detector parts (OM, cables, LCM, etc.) is then available. The locations in the detector of these parts are stored in the database. This allows to trace back its history in case of problems.

Detector setup

The description of the detector setup is stored in the following database tables:

host	IP address | process
address	IP address | container
container	container | device | ID | status
device	device | MODBUS address | parameter
mapping	container | device | parameter | object

The **host** table is used by the **RunControl** programme to launch the necessary processes on the corresponding host CPUs and to keep track of the state of each process. The **address** table links the identifier of each processor to its geographical location in the detector (e.g. string and storey). The locations of the hardware devices in the detector are listed in the **container** table. The status column allows to specify the state of each device in the system (e.g. active, sleep or dead). The **device** table links the hardware device to its MODBUS address. It is assumed that this address is independent of the location of the device in the detector (container). The **mapping** table maps the ARS mother board (**ARS_MB**) identifier (1-3) to the PMT identifier (1-3). The parameter column is used to map the high voltage channel identifier of the optical module board (COMPASS_MB) to the PMT identifier. These tables are created and modified using the **Oracle8*i* GUI**.

Detector configuration

The configuration of the detector is stored in the following database tables:

settings	label | time | container | device | parameter | value | min. | max.
monitor	label | time | container | device | frequency | min. | max.
trigger	label | time | parameter | value
filter	label | time | parameter | value
conversion	device | ID | time | parameter | offset | slope | min. | max.

The labels in these tables refer to a complete set of parameter values (e.g.``current''). The min. and max. columns in the **settings** table specify the limits of these values. The frequency, min. and max. columns in the **monitor** table specify the frequency of reading that parameter and the range of acceptable values. The values of various columns can be changed using the **SettingsEdit**, **MonitorEdit**, **TriggerEdit** and **FilterEdit** GUIs. The **conversion** table is used to convert the software values in the database tables to (integer) values for the corresponding hardware device. The conversion parameters are measured beforehand. The min. and max. columns specify the minimal and maximal (integer) values corresponding to the number of (hardware) bits. The tables are created and modified using the **Oracle8*i* GUI**.

Detector operation

The parameter values read regularly during operation of the detector are stored in the following database tables. With a typical frequency of about 1 min^{-1}, 100 GB of data are stored per year.

readings	RUN | RTS | container | device | parameter | value | status
acoustic	RUN | RTS | container | cycle | time | amplitude | status
clock	RUN | GPS time | container | delay | status

The run number and the values of the RTS counter are used to link off-line the measured values to the physics data.

Detector alignment and calibration

The detector alignment and calibration is stored in the following database tables:

alignment	label | time | key | *(x,y,z)* | status
calibration	label | time | container | PMT | ARS | t_0 | t_1 | offset | slope | status
alignment	amplitude | label | time | container | PMT | ARS | gain | status

These tables are used for the off-line analysis of the data. The alignment is determined from the

acoustic data table using the **DetAlign** programme. The (time) calibration is determined from the data taken during the flashing of optical beacons using the **TimeCalib** programme and the amplitude (calibration) from the (background) data using the **GainCalib** programme. The labels in the **alignment** and **calibration** tables refer to the run number (with an optional extension).

Data taking

The data taking actions are stored in the following database tables:

runoptions	RUN | GPS time | table name | entry label
operator	RUN | GPS time | event | counter
runlist	RUN | GPS time | # records | size | status
tapelist	RUN | tape | status
logbook	date | time | RUN | label | text

The **runoptions** table keeps track of the selected database tables used for the configuration of the detector, the settings of the trigger and the parameters of the filter algorithm. The **operator** table associates the GPS time to the events generated by the operator (*on, start, pause, continue, stop* and *off*) and the signals distributed by the clock system (SOR and RTS). The counter column refers to the number of occurrences of a particular events during a data taking run. The **runlist** and **tapelist** tables contain information from the **dataWriter** and **tapeWriter** processes, respectively. The **logbook** table is used by the **LogBook GUI**.

Annex A: Detector setting parameters

container	device	parameter	unit	min.	max.	step
LCM	COMPASS_MB	HV_PMT1	V	0	2000	1
LCM	COMPASS_MB	HV_PMT2	V	0	2000	1
LCM	COMPASS_MB	HV_PMT3	V	0	2000	1
LCM	ARS_MB*i*[1]	Nd[2]	Hex	0x0	0x7F	1
LCM	ARS_MB*i*	En_clk_out	unit	0	1	1
LCM	ARS_MB*i*	Reset_pll	unit	0	1	1
CML	ARS_MB*i*	Clc2	unit	0	1	1
LCM	ARS_MB*i*	En_ext_cap	unit	0	1	1
LCM	ARS_MB*i*	Link_clk_ref	unit	0	1	1
LCM	ARS_MB*i*	ATWR	unit	0	1	1
LCM	ARS_MB*i*	Pll	Hex	0x0	0xF	1
LCM	ARS_MB*i*	Pll_v1	Hex	0x0	0xF	1
LCM	ARS_MB*i*	Pll_v2	Hex	0x0	0xF	1
LCM	ARS_MB*i*	En_inb_aclk	unit	0	1	1
LCM	ARS_MB*i*	En_inb_dyn2	unit	0	1	1
LCM	ARS_MB*i*	En_inb_dyn1	unit	0	1	1
LCM	ARS_MB*i*	En_inb_an	unit	0	1	1
LCM	ARS_MB*i*	OPA_bias	Hex	0x0	0xF	1
LCM	ARS_MB*i*	En_dcel	unit	0	1	1
LCM	ARS_MB*i*	TVC_bias	Hex	0x0	0xF	1
LCM	ARS_MB*i*	Dyn_th	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	Int_bias	Hex	0x0	0xF	1
LCM	ARS_MB*i*	Trig0_th	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	PSD_th	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	Spe_clk	Hex	0x0	0xF	1
LCM	ARS_MB*i*	Spe_gate	Hex	0x0	0xF	1
LCM	ARS_MB*i*	Spe_pipe	Hex	0x0	0xF	1
LCM	ARS_MB*i*	Scale_wav	Hex	0x0	0x3	1
LCM	ARS_MB*i*	Delay_way	Hex	0x0	0xF	1
LCM	ARS_MB*i*	Sel_pled_clk	unit	0	1	1
LCM	ARS_MB*i*	En_acq	unit	0	1	1
LCM	ARS_MB*i*	Ev_sc	unit	0	1	1
LCM	ARS_MB*i*	En_edge	unit	0	1	1
LCM	ARS_MB*i*	All_spe	unit	0	1	1
LCM	ARS_MB*i*	All_wave	unit	0	1	1
LCM	ARS_MB*i*	PSD_tot	Hex	0x0	0x1F	1

LCM	ARS_MB*i*	CRM_pc	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	CRM_sel_clk	unit	0	1	1
LCM	ARS_MB*i*	CRM_w	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	Burst	unit	0	1	1
LCM	ARS_MB*i*	Sel_pulse	unit	0	1	1
LCM	ARS_MB*i*	En_pulser	unit	0	1	1
LCM	ARS_MB*i*	Readme_p	unit	0	1	1
LCM	ARS_MB*i*	Flag_ware_p	unit	0	1	1
LCM	ARS_MB*i*	dyn_p	unit	0	1	1
LCM	ARS_MB*i*	enf_p	unit	0	1	1
LCM	ARS_MB*i*	resf_p	unit	0	1	1
LCM	ARS_MB*i*	emptyb_p	unit	0	1	1
LCM	ARS_MB*i*	crmf_p	unit	0	1	1
LCM	ARS_MB*i*	Dcl_bias	Hex	0x0	0x7	1
LCM	ARS_MB*i*	Acc_t2	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	Wacc_t2	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	Scale_acc_t2	unit	0	1	1
LCM	ARS_MB*i*	Adj_ac_t2	Hex	0x0	0x3	1
LCM	ARS_MB*i*	Delay_t2	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	Delay_t1	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	No_t1d	unit	0	1	1
LCM	ARS_MB*i*	No_t2d	unit	0	1	1
LCM	ARS_MB*i*	All_t1b	unit	0	1	1
LCM	ARS_MB*i*	T0_width	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	Test_an	Hex	0x0	0x3	1
LCM	ARS_MB*i*	Padc	unit	0	1	1
LCM	ARS_MB*i*	Test	unit	0	1	1
LCM	ARS_MB*i*	Test_sel	Hex	0x0	0xF	1
LCM	ARS_MB*i*	AD2_b2b	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD2_b1b	Hex	0x0	0x1 F	1
LCM	ARS_MB*i*	AD2_b0b	Hex	0x0	0x1 F	1
LCM	ARS_MB*i*	AD2_b2l	Hex	0x0	0x1 F	1
LCM	ARS_MB*i*	AD2_b1l	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD2_b0l	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b2b	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b1b	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b0b	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b2l	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b1l	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b0l	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	Test_dg	Hex	0x0	0x3	1
LCM	ARS_MB*i*	Clsw	unit	0	1	1

LCM	ARS_MB*i*	CRM_pc	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	CRM_sel_clk	unit	0	1	1
LCM	ARS_MB*i*	CRM_w	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	Burst	unit	0	1	1
LCM	ARS_MB*i*	Sel_pulse	unit	0	1	1
LCM	ARS_MB*i*	En_pulser	unit	0	1	1
LCM	ARS_MB*i*	Readme_p	unit	0	1	1
LCM	ARS_MB*i*	Flag_ware_p	unit	0	1	1
LCM	ARS_MB*i*	dyn_p	unit	0	1	1
LCM	ARS_MB*i*	enf_p	unit	0	1	1
LCM	ARS_MB*i*	resf_p	unit	0	1	1
LCM	ARS_MB*i*	emptyb_p	unit	0	1	1
LCM	ARS_MB*i*	crmf_p	unit	0	1	1
LCM	ARS_MB*i*	Dcl_bias	Hex	0x0	0x7	1
LCM	ARS_MB*i*	Acc_t2	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	Wacc_t2	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	Scale_acc_t2	unit	0	1	1
LCM	ARS_MB*i*	Adj_ac_t2	Hex	0x0	0x3	1
LCM	ARS_MB*i*	Delay_t2	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	Delay_t1	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	No_t1d	unit	0	1	1
LCM	ARS_MB*i*	No_t2d	unit	0	1	1
LCM	ARS_MB*i*	All_t1b	unit	0	1	1
LCM	ARS_MB*i*	T0_width	Hex	0x0	0xFF	1
LCM	ARS_MB*i*	Test_an	Hex	0x0	0x3	1
LCM	ARS_MB*i*	Padc	unit	0	1	1
LCM	ARS_MB*i*	Test	unit	0	1	1
LCM	ARS_MB*i*	Test_sel	Hex	0x0	0xF	1
LCM	ARS_MB*i*	AD2_b2b	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD2_b1b	Hex	0x0	0x1 F	1
LCM	ARS_MB*i*	AD2_b0b	Hex	0x0	0x1 F	1
LCM	ARS_MB*i*	AD2_b2l	Hex	0x0	0x1 F	1
LCM	ARS_MB*i*	AD2_b1l	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD2_b0l	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b2b	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b1b	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b0b	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b2l	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b1l	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	AD1_b0l	Hex	0x0	0x1F	1
LCM	ARS_MB*i*	Test_dg	Hex	0x0	0x3	1
LCM	ARS_MB*i*	Clsw	unit	0	1	1

LCM	LCM_TRIG	L1_enable	unit	0	1	1
LCM	LCM_TRIG	L2_Istorey_enable	unit	0	1	1
LCM	LCM_TRIG	L2_2storey_enable	unit	0	1	1
LCM	LCM_TRIG	L1_config	unit	1	4	1
LCM	LCM_TRIG	L2_config	unit	1	4	1
LCM	LCM_TRIG	Delay_L2trig	ns	10	200	1
LCM	LCM_TRIG	Delay_LCMJB	ns	0	5000	10
LCM	LCM_TRIG	TwT1	ns	10	200	1
LCM	OB	Intensity	V	7	24	1
LCM	ACOUST_Rx	Nel[3]	unit	1	20	1
LCM	ACOUST_Rx	F_Rx[3]	Hz	40000	65000	1
LCM	ACOUST_Rx	autoGain[3]	unit	0	1	1
LCM	ACOUST_Rx	anaGain[3]	dB	0	80	1
LCM	ACOUST_Rx	digGain[3]	dB	-20	+20	1
LCM	ACOUST_Rx	th[3]	unit	0x0000	65535	1
LCM	ACOUST_Rx	maxDD[3]	ms	0	10000	1
LCM	ACOUST_Rx	tw1[3]	µs	0	1000000	1
LCM	ACOUST_Rx	tw2[3]	µs	0	1000000	1
SCM	ACOUST_Rx	Nel[3]	unit	1	20	1
SCM	ACOUST_Rx	DSP[3]	unit	1	2	1
SCM	ACOUST_Rx	F_Rx[3]	Hz	40000	65000	1
SCM	ACOUST_Rx	autoGain[3]	unit	0	1	1
SCM	ACOUST_Rx	anaGain[3]	dB	0	80	1
SCM	ACOUST_Rx	digGain[3]	dB	-20	+20	1
SCM	ACOUST_Rx	th[3]	unit	0x0000	65535	1
SCM	ACOUST_Rx	maxDD[3]	ms	0	10000	1
SCM	ACOUST_Rx	tw1[3]	µs	0	1000000	1
SCM	ACOUST_Rx	tw2[3]	µs	0	1000000	1
SCM	ACOUST_Rx	F_Tx[4]	Hz	40000	65000	1
SCM	ACOUST_Rx	level[4]	unit	0	1	1
SCM	ACOUST_Rx	delay[4]	µs	1	250	1
SCM	ACOUST_Rx	duration[4]	µs	1	100	1
ON_CLOCK	ACOUST_CONTROL	delay	ms	10	1000	10
ON_CLOCK	ACOUST_CONTROL	slow_sync	s	1	600	1
ON_CLOCK	ACOUST_CONTROL	sync	s	1	20	1
ON_CLOCK	ACOUST_CONTROL	fast_sync	ms	100	10000	10

[1] The index **i** refers to the PMT or the optical beacon.
[2] The (actual) parameter setting can be different for the ARSs on the same motherboard (device).
[3] There is one parameter per elementary reception cycle.
[4] There is one parameter per elementary emission cycle.

Table A.1: Specifications for the **settings** table.

Annex B: Detector monitor parameters

container	device	parameter(s)	frequency
LCM	POWER_BOX	V48, V48s, V12, V5.5, V5, V3.3, V2.5, V1.8, I48, I48s, I12, I5.5, I5, I3.3, I2.5, I1.8	1 min^{-1}
LCM	LCM_DAQ/SC	temp	1 min^{-1}
LCM	COMPASS_MB	cap, pitch, roll, Bx, By, Bz, temp, error, checksum	2 min^{-1}
LCM	COMPASS_MB	humidity, Temp1, Temp2, T_ARS_MB1, T_ARS_MB2, T_ARS_MB3, (T_ARS_MB4), HV_PMT1, HV_PMT2, HV_PMT3	12 $hour^{-1}$
LCM	LCM_CLOCK	V1	1 min^{-1}
LCM	LCM_BIDICON	V1	1 min^{-1}
LCM	LCM_BIDIANT2	V1	1 min^{-1}
LCM	LCM_TRIG	L0_pmt1, L0_pmt2, L0_pmt3, LX_pmt12, LX_pmt13, LX_pmt23, L1U, L1D and L2	1 s^{-1}
LCM	LCM_BIDITRIG	V1	1 min^{-1}
LCM	LCM_SWITCH	V1	1 min^{-1}
LCM	LCM_DWDM	temp, Ibias, Imodu, I_TEC, V_TEC, power, Tlock, V1 and V2	6 $hour^{-1}$
LCM	ACOUST_Rx	Elementary cycle, time and amplitude, positioning cycle	1 min^{-1}
SCM	COMPASS_MB	cap, pitch, roll, Bx, By, Bz, temp, error, checksum	12 $hour^{-1}$
SCM	COMPASS_MB	humidity, Temp1, Temp2	12 $hour^{-1}$
SCM	POWER_BOX	V48, V48s, V12, V5.5, V5, V3.3, V2.5, V1.8, I48, I48s, I12, I5.5, I5, I3.3, I2.5, I1.8	1 min^{-1}
SCM	ACOUST_RxTx	Elementary cycle, time and amplitude, positioning cycle	1 min^{-1}
SCM	ACOUST_PRESS	pressure	1 min^{-1}
SCM	ACOUST_SVEL	velocity	1 min^{-1}
SCM	ACOUST_SVEL-CTD	velocity, conductivity, temp and pressure	1 min^{-1}
SCM	ACOUST_CTD	conductivity, temp and depth	1 min^{-1}

SPM	POWER_BOX	Vin, V1out, V2out, V3out, V4out, V5out, V6out, I1out, I2out, I3out, I4out, I5out, I6out, T1, T2, T3, IACleak, IDCleak	1 min^{-1}

Table B.1: Specifications for the **monitor** table.

Annex C: Detector conversion parameters

container	device	parameter	offset	slope	min.	max.
LCM	COMPASS_MB	HV_PMT1	0	1/9.77	0	4095
LCM	COMPASS_MB	HV_PMT2	0	1/9.77	0	4095
LCM	COMPASS_MB	HV_PMT3	0	1/9.77	0	4095

Table C.1: Specifications for the **conversion** table.

Annex D: List of References

1. http://www.nikhef.nl/user/mjg/CHSM/pjl-thesis.pdf
2. C. Olivetto, ``UNIV1 card'', ANTARES-ELEC/2000-03
3. D. Lachartre, ``*ARS1* Analogue Ring Sampler'', ANTARES-ELEC/2000-06
4. http://www.nikhef.nl/user/ruud/HTML/matra.html'
5. http://www.nikhef.nl/user/mjg/ControlHost
6. J.Carr and A. Pohl, ``Another new trigger for ANTARES'', ANTARES-ELEC/1999-03
   R. Bland et al., ``trigger calculations for the ANTARES CDR'', ANTARES-ELEC/2000-04
7. N. Palanque-Delabrouille, ``Optical background measurements'', ANTARES-SITE/1998-02
8. P. Coyle ``Specification for the Antares Offshore Trigger'', ANTARES-ELEC/2000-07
   F. Rethore, ``Antares Trigger Card'', in preparation
9. http://www.brandywinecomm.com
10. V. Bertin et al., ``Conclusions of the Time Calibration Committee'', ANTARES-CALI/2000-07