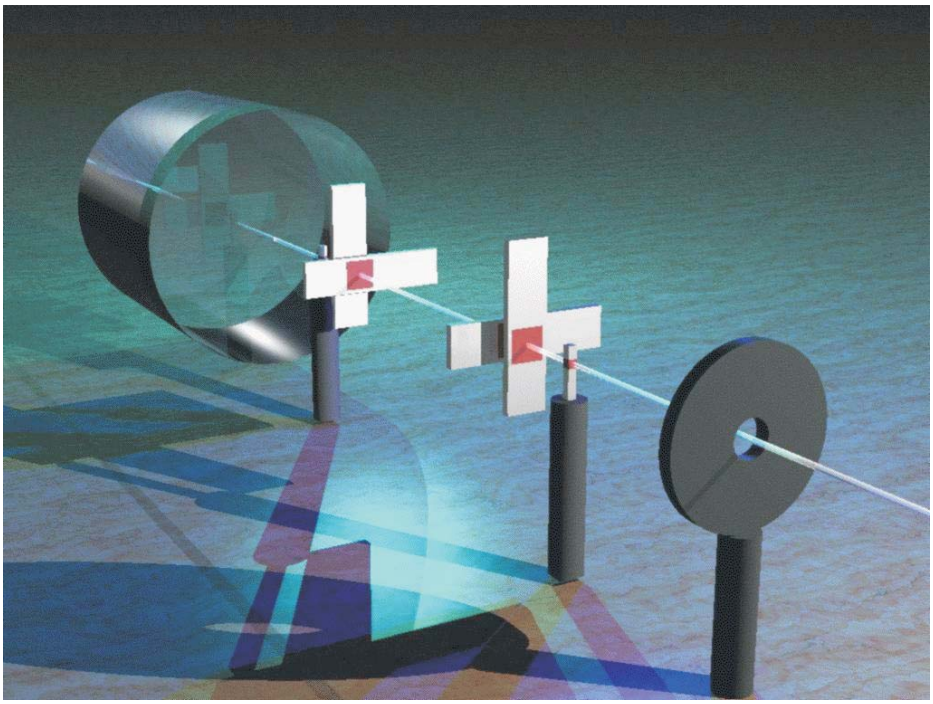
The Bonn ATLAS Telescope

Johannes Treis

October 29, 2001



Artist's view of a beam telescope

Abstract

A detailed description of the Bonn ATLAS Telescope (BAT) system including software and hardware is given. The BAT is a reference instrument for charged particle tracking. With its four modules, each containing a double sided silicon strip detector, the BAT provides a quadruple of space points used for trace reconstruction of a charged particle.

In contrast to existing telescope systems, the BAT is designed for fully PC-based operation. On-board digitization of analog data and intelligent data preprocessing logic are integrated in each module to perform on-board hit detection and zero suppression. Thus, the system can process an event rate about 15 times larger than the event rate achieved with common VME-based systems.

The BAT can support almost any kind of DUT. The benefits of the BAT, namely high readout speed and data preprocessing, can be carried over to the DUT via a specific interface. Additionally, the system is optimized for low noise operation and the relatively low beam energies provided by the ELSA testbeam facility in Bonn.

Integration of VME components into the BAT readout is possible. Moreover, an interface for VME based operation of the entire telescope system exists.

Contents

List of Figures

List of Tables

1 Introduction

When developing detector systems for ionizing radiation, testing a new device under conditions very similar to the experiment is a common problem. As far as high energy physics is concerned, the opportunity to undertake these tests is offered by testbeam facilities, which create an environment very similar to the surrounding of a high energy physics experiment. Such a testbeam facility is usually located close to a larger accelerator, extracts parts of the intensity of the main (primary) beam and converts it into a secondary beam some of whose properties, e.g. particle contents, intensity and particle energy, are well-defined.
However, when examining the characteristics of position sensitive detectors[1] properties of the secondary beam are not as well-known as the experimentalist would like them to be. Although a testbeam facility usually offers a variety of possibilities to steer the beam and influence the beam profile, the special kind of information which is important for this kind of test is usually not provided. The testbeam is not a continuous beam, of course, but consists of a stream of high energy particles. The passing of a single beam particle through the detector which is to be tested[2] is called an *event*. What is needed now is an information which allows a comparison between the event data acquired from the DUT and the "event reality". The "event reality" is the *penetration point*, the point, where in this event the particle crossed the sensitive area of the DUT and its angle of incidence. As this information is not provided by the testbeam facility, it has to be obtained elsewhere, and at this point a beam telescope enters the stage.
A beam telescope is a detector system which can measure the *track* of an incident ionizing particle, that is, the exact position and angle with respect to a plane of reference [3], with some accuracy. From this data the penetration point on the DUT and its error can easily be calculated if the position of the DUT with respect to the POR is known. Figure 1 shows the concept of such a particle tracking detector.

Although this is the primary task of a beam telescope system, there are other requirements which arise e.g. from the fact that the error on the track measurement, and therefore on the penetration point forecast, has to be as small as possible, that the event rate, which is the number of events which can be processed, has to be high and, of course, that the system has to be

[1] Especially their spatial resolution is what physicists are interested in here.

[2] The sensor and / or the detector system whose properties are to be determined is in the following referred to as *device under test*, abbr. *DUT*.

[3] This plane of reference is in the following referred to as *POR*.

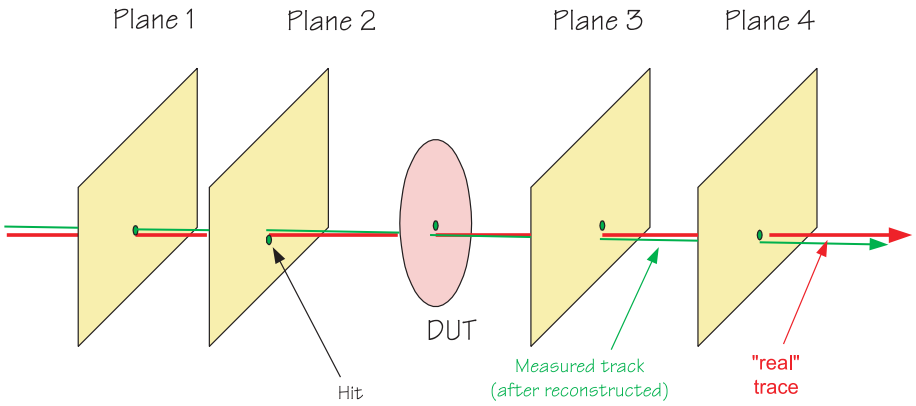


Figure 1: Concept of a particle tracker consisting of a number of discrete detector planes. The passing particle generates a signal in each plane, which allows the reconstruction of the penetration point with some accuracy. Evaluating the data from all detector planes, the particle track can be obtained with a simple linear regression. Of course, the position of the detector planes with respect to each other (alignment) has to be known as precise as possible. The error for this interpolated track corresponds to the error of the regression. If the alignment of the DUT with respect to the telescope is known, the DUT data and the interpolated track based on telescope data can be compared.

efficient, robust and easy to operate. In the following, some of the demands on such a telescope system are listed:

- The beam telescope itself has to influence the beam in the slightest possible way. So, the track of the incident particle can be measured as precisely as possible, and the penetration point can be predicted as accurate as possible. Influence on the beam can occur e.g. due to coulomb multiple scattering at the atoms of the material of which the telescope consists, so it is a good idea to put as little material into the beamline as possible[4] and to use material with low atom number to lower the cross section for multiple scattering to the lowest possible value.

- For a detailed analysis of sensor properties a lot of statistics is needed. Especially in case of semiconductor sensors, with pixel sizes at the order of 250 square microns or even smaller, it takes a long time to collect enough statistics to be able to characterize the properties of every single pixel cell in a satisfying way. The time it takes to collect this amount

[4] This is the reason why people say that they would favor a telescope "that isn't there at all".

of data depends on the readout speed of the DUT and the telescope on one side and on the beam intensity on the other. So the resulting demand to the telescope system is to allow an event rate that high that the only limit is given either by the DUT readout or the beam intensity. If this demand cannot be met, the telescope readout should at least be as fast as possible.

- One of the important parameters developers are interested in is the DUT *efficiency*. The detector efficiency is the fraction of events, in which the DUT was able to detect the incident particle. To measure the efficiency, one has to determine the real number of particles passing the DUT and this is only possible if the reference counter used (i.e. the telescope) is ideally 100% efficient, but at least much more efficient than the DUT.

- As the demands to as well software as hardware of the telescope system can be very different depending on which kind of DUT is to be examined, a multi-purpose-system has to offer a flexible, well-defined hardware and software interface for connecting a DUT to the system and including it in the readout chain. Besides, a possibility to connect the system to standard electronic components used for measurement and data acquisition in high energy physics (e.g. VME modules) should also be provided.

- As such a system will be used many times by many different people, the system has to be robust, easy to handle and easy to operate. The setup needed for such a system has to be as simple as possible to avoid problems in preparing the measurements. The system may travel to a lot of different testbeam facilities, so the equipment needed for full operation should be reduced to the minimum to preserve mobility.

Of course, this list is not complete. The Bonn ATLAS telescope was designed to meet most of these requirements, and where the requirements were not met, the performance compared to older systems was significantly improved.

2 System overview

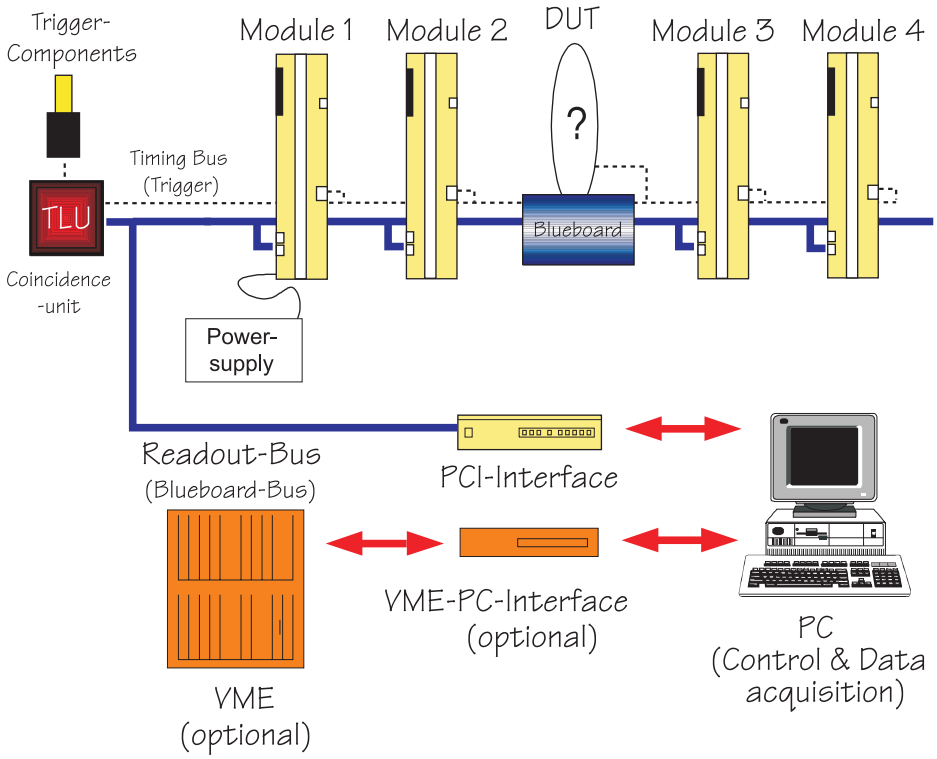


Figure 2: Schematic overview about the BAT setup. The PC controls the whole system. The blueboard bus only transfers digital data and commands. Each module and DUT has its own power supply. The connection between PC and VME is made via a commercial PC-VME interface. This is only needed if VME components are to be included into the readout chain.

An overview about the whole BAT system is shown in fig. 2 The BAT consists of four planes of double sided silicon strip detectors each providing 640 channels / side. Each plane is read out by VA 2 ICs provided by IDE AS, Oslo. Directly connected to the front end ICs are complete analog and digital signal processing circuits enabling signal digitization and data preprocessing. Detector, front-end electronics and signal processing and digitization logic are mounted in a solid aluminum case. This is what is called a BAT module. Each module has its own power supply. It provides not only the supply voltages for the analog and digital circuits inside the module, but also the high voltage detector bias.
The system is fully PC based. The whole control, data acquisition and data storage software runs on a PC operating under windows NT / 9x OS. The connection to the hardware is made via the BB-PCI interface card provided

by Silicon Solutions. The BB, the Blueboard Bus, is a proprietary bus standard. All modules are connected to one another and the DAQ PC over this BB. The BB connections are shown in figure 3.

The BB is purely digital. Only digital signals and commands are exchanged between the PC and the connected modules. Thus, there is no need to transmit analog signals over long cables to external (e.g. VME) ADCs[5], which improves the system's setup as well as the system's analog performance.

The trigger for the system has to be provided externally, e.g. by a trigger scintillator coincidence. This trigger signal is not to be fed directly into the BAT modules. Another module connected to the BB, the TLU (trigger logic unit), serves as trigger fan-out and as trigger controlling stage. For, if properly configured, the modules act completely independent from any external sequencer logic, such a stage is needed to enforce synchronous triggers and do a controlled start and end of run. The modules themselves receive and process the trigger. The data obtained in this way is stored in the module-internal memory blocks (FIFO). The module itself sends an IRQ either if one event has been successfully processed (event interrupt) or if a certain amount of data gathered in the modules FIFO (data interrupt, buffered mode). For various reasons, the buffered mode is much faster than the event IRQ mode. All that remains to the DAQ software is to read back the data from the modules, assemble them to complete events and store them somewhere. A part of the data acquired in this way can be used to display online monitoring histograms, which show the systems actual performance and allow the user to check if everything is going well.

The system offers two ways to connect devices under test (DUTs). VME based devices for example can be connected using a common VME-PC interface. This has been successfully done for several testbeam periods the BAT

[5]This is done in most of the telescope systems currently used.

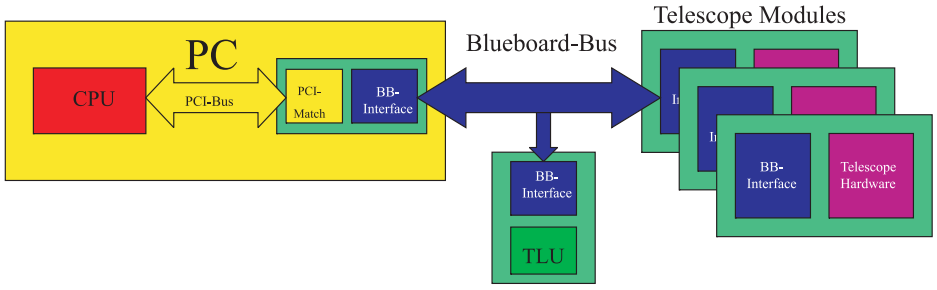


Figure 3: BAT module interconnection using the BB. The modules are all tied daisy chained to the same bus cable. The TLU has to be last module on the bus.

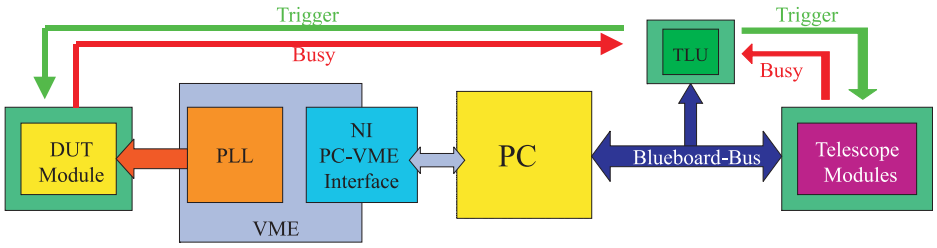


Figure 4: How to include a VME based DUT. Access to the VME resources is gained with e.g. a common PC-VME interface. Problems arise if the VME module is not capable of data buffering or IRQ generation.

was used in. The problem here is, that it is difficult to hand an IRQ on the VME bus directly to the CPU (at least for the VME-PC interfaces commonly used). So, an event-by-event readback has to be made, which significantly lowers the system's performance in speed[6]. The setup necessary is shown in figure 4.

The other possibility is to connect the DUT module directly to the BB. Here, the digital logic of a BAT module (DTC) can be used as a ready-to-go, easy-to-handle interface, in which with little effort concerning FPGA programming the readout functions for the DUT can be implemented. This setup is shown in figure 5. Sometimes an analog interface card has to be made, which provides supply and bias voltages to the DUT (thus replacing the BAT analog circuits). But the great advantage is that the module created in this way can easily be integrated on the BB and in the readout

[6]But even in this case, the performance improves compared to common systems.

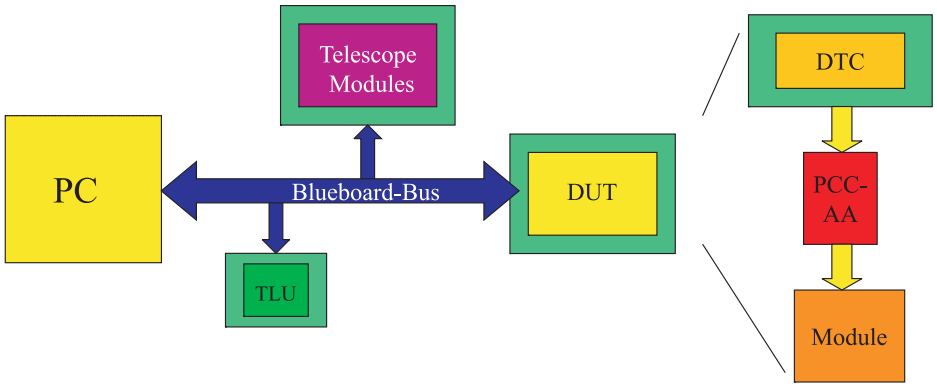


Figure 5: How to include a BB based DUT. The module is simply tied to the BB as well. If e.g. the DTC card is used as interface to the BB, the insertion of the module to the software will be easy as well.

software. It also supports data buffering and IRQs, so the system's high speed capabilities can fully be used. This setup is shown in fig. 6.

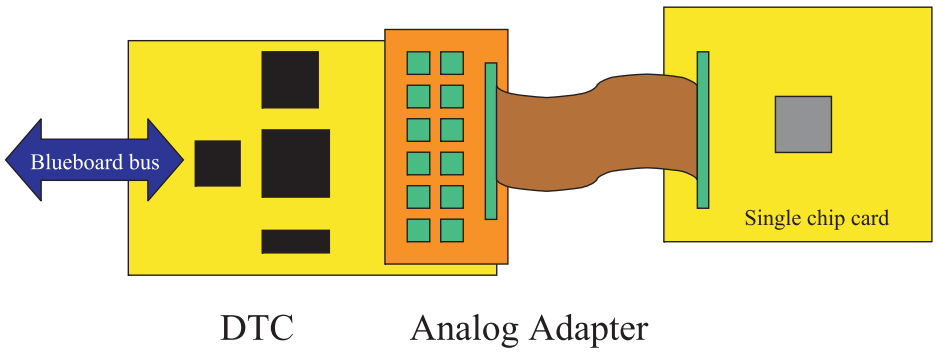


Figure 6: Setup of a BB based DUT using the BAT DTC as interface to the BB. The analog adapter replaces the ATC card used in the BAT modules. This is the easiest way to include DUTs, also from the software point of view.

3 The Blueboard bus

All BAT modules, the TLU and sometimes also DUTs are connected to the DAQ PC via the Blueboard Bus (BB). This is a proprietary bus standard which has been developed by Silicon Solutions, a small company from Cologne. It has some similarities to SCSI, transferring differential signals over long distances with great reliability. The BB also offers flexible functions for system configuration, system administration and IRQ handling. The BB software package provided by Silicon Solutions contains debugging tools, device drivers and software libraries which offer a very transparent and easy-to-operate software interface for the operation of any kind of module. The hardware interface is also well-defined and can be adapted to the users needs with little or even no hardware effort, depending on the users requirements. In the following, the BB basics are described as far as it is necessary to understand telescope operation.

3.1 BB basics.

The BB is designed to be operated from a PC platform with a PCI bus. The connection to the BB is made via the BB-PCI interface card. This card contains a Xilinx XC 3142 FPGA, which in the following be referred to as BB FPGA, and an AMCC PCI matchmaker IC. The BB FPGA has controls the interface and timing to the BB, the AMCC controls all PCI access functions. An image of the BB-PCI interface card is shown in fig. 7. The BB itself is connected to this card using 50 pol. Centronics standard connectors. The modules can either be connected using standard IDC connectors, or, if some

Figure 7: Picture of the BB-PCI interface card. A free standard PCI slot is needed for the installation. Supported OS are windows 95 / 98 and windows NT 4.0. Drivers for windows 2000 are not yet available.

effort is made to design a case for the modules (e.g. for the BAT modules), some customized connectors can be used[7]. The bus itself is made up from 50 pol. twisted pair flat ribbon cable[8].
The BB consists of a bidirectional 8 bit data bus, a 3 bit address bus and several command and control signals. The most important of these signals are:

- Mode / Data (M/D)
- Read / Write (R/W)
- DWORD
- Strobe (STR)
- Acknowledge (ACK)

The data bus is either driven by the busmaster during a write access, displaying the data which is to be written, or by the accessed module during a read access, displaying the data the module has stored in its selected register. The 3 bit module address bus is controlled by the busmaster and shows the address of the module which is currently going to be accessed. The R/W line decides whether the next access will write into a register in the destination module or if a value is to be read from it. The BB supports two kinds of accesses: byte accesses transferring 8 bit each, and DWORD accesses transferring 32 bit in 4 packages of 8 bit each. The DWORD line shows which kind of access is currently performed. A byte access can occur either to the central mode address register of a connected module, selecting a register which is to be accessed next (this is a mode access), or to the selected register itself (this is a data access). The M/D line decides whether the mode address register or the register selected with the mode address register value is to be accessed. As the mode address register is usually only 8 bit wide, there is no such thing as a MODE-DWORD access. A strobe is sent during a write data access to make the accessed module accept the data currently valid on the data bus. During a read access a strobe is also sent by the busmaster, but in this case it is used to make the accessed module put the data of its selected register on the data bus. If this process is finished, the module sends an ACK signal, which indicates the busmaster, that the data currently valid on the bus is the data which is to be read back. The BB-PCI card is the only busmaster on the BB. It has the BB address 0 per default. As there are 7

[7] For the BAT modules we decided to use the Centronics standard connectors.
[8] Twisted pair is necessary because the BB transmits differential signals.

BB addresses left, up to 7 devices can be connected to the BB. In case this is not enough, more BB PCI interface card can be added to the system, if there is an appropriate number of free PCI slots. This feature has not been used for the BAT.
As described above, there are 6 different kinds of accesses to the BB:

1. Write mode
2. Write data
3. Read mode
4. Read data
5. Write DWORD
6. Read DWORD

Each of these accesses can be performed to each module connected to the BB. The connection on software side is made via the C++ software units provided by Silicon Solutions. At startup the software automatically initializes the interface hardware. Afterwards, the software scans the entire BB address range (from 0 to 7) to find out if there are modules connected to the BB and which kind of module it is. This is done by attempting to read the implementation ID, which can be done by doing a read data to MA 0[9]. If for a certain BB address no module is connected, there won't be an ACK signal indicating valid data on the bus, so the software knows that there is no module connected to the bus bearing this address. If a module is connected, a read access to MA 0 will yield not only an ACK answer to the strobe, but also an implementation ID. The 3 MSBs indicate the type of the connected module (there are different kinds of them), the 5 LSBs contain the firmware revision number. Module types defined so far are listed in table 1. Additionally, the figures 8 and 9 give an overview about the access timings for the different access types.

After detecting a module of a certain kind, the rest of the software is told, that a module of a certain kind exist, thus controls, windows and stuff can be prepared correctly. Additionally, certain pointers are prepared, one for each possible BB access, which, due to memory mapping, allow an access to the BB PCI interface, enabling the user to perform a certain access by simply accessing the address given in a certain pointer for this module. The

[9]Per convention, this has to be the MA for the ID readback in all IFPGAs.

Module type	ID	
PCI CARD	0	The BB PCI interface itself.
PCI REC	1	The BAT module DTC implementation.
TESTBOARD	2	The standard Silicon Solutions testboard. May be used as a DUT interface to the BB sometimes.
TLU	4	Trigger logic unit. Trigger controlling stage for telescope operation.
FE-DUT	5	Example for DUT integration. Based on the DTC firmware. Allows integration of an ATLAS single chip into the telescope environment.
SPECIAL	6	ID for customized devices. Implementation case depending.

Table 1: List of module types defined so far. The number given corresponds to the value of the 3 MSBs of the module firmware implementation ID, which can be read back by accessing the module's MA 0.

following pointers are initialized and automatically made available by the software:

1. dword
2. data
3. mode

Please note that the pointers themselves should remain unchanged. An access like $*mode = value$ writes the MA register of the module for which the pointer is valid, $value = *mode$ stores the value of the MA register in the variable *value* and so on, but an access like $mode = value$ is to be avoided, for in 99 of 100 cases it results in confusion, because the pointer to the hardware resources gets lost and the system has to be re-initialized. As will be explained in the section about the software, these pointers are data members of the class `TReceiver`, which is, or should be, the parent class for all properly programmed BB modules. A complete description of the software implementation would exceed the scope of this manual. What should be kept in mind is that performing a certain BB access to a certain module can simply be done by accessing the address stored in the corresponding pointer for this module. A list of all accesses is given in table 2. Please note that

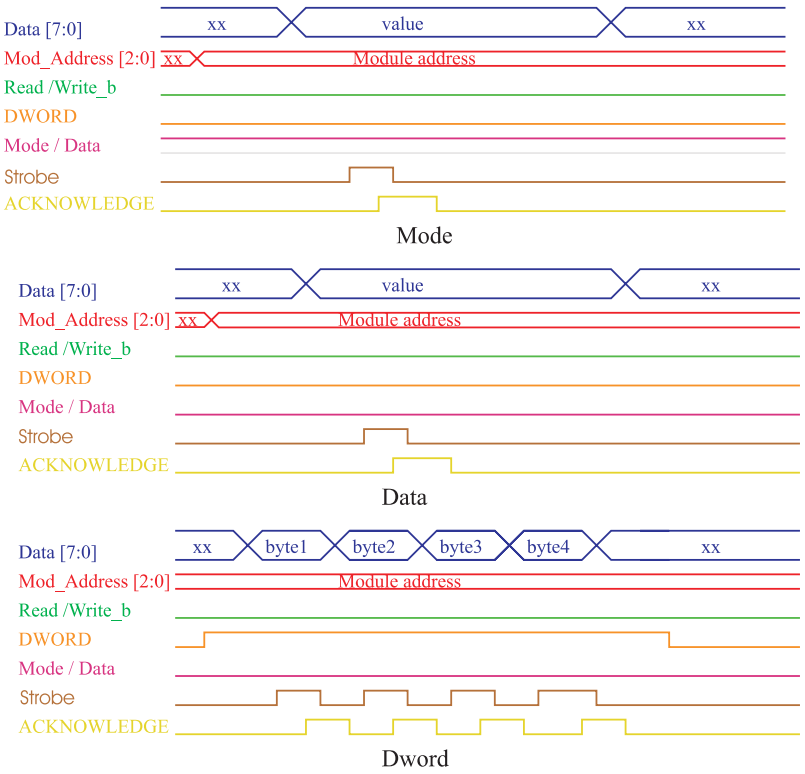


Figure 8: Signal timing on the BB (write accesses). Timing is not to scale. The WR_ CLK is a 16 MHz clock.

for a DWORD access a 32 bit value has to be made available. Performing a DWORD access to a byte register will yield in the best case a value which consists of four times the same byte.

Another thing is also important with the DWORD access. While for a byte access each time the BB address for the module which is going to be accessed is put on the bus, this is not the case for the DWORD access[10]. **The BB address is not changed for a DWORD access.** So, before a DWORD access to a certain module, care has to be taken to perform either a mode or a (byte) data access to exactly this module to have the desired module address applied. Otherwise the access might address a completely different module. But as in most cases before starting a DWORD access a MA has to be written to select the register, to which the DWORD access is to be performed, this condition is not very difficult to meet. Care has to be taken only when BB accesses can be performed by different threads. If e.g. during a chain of DWORD accesses performed by thread A on module 1 thread B

[10]This is due to speed the access up; writing the address takes almost as long as an entire data (byte) access.

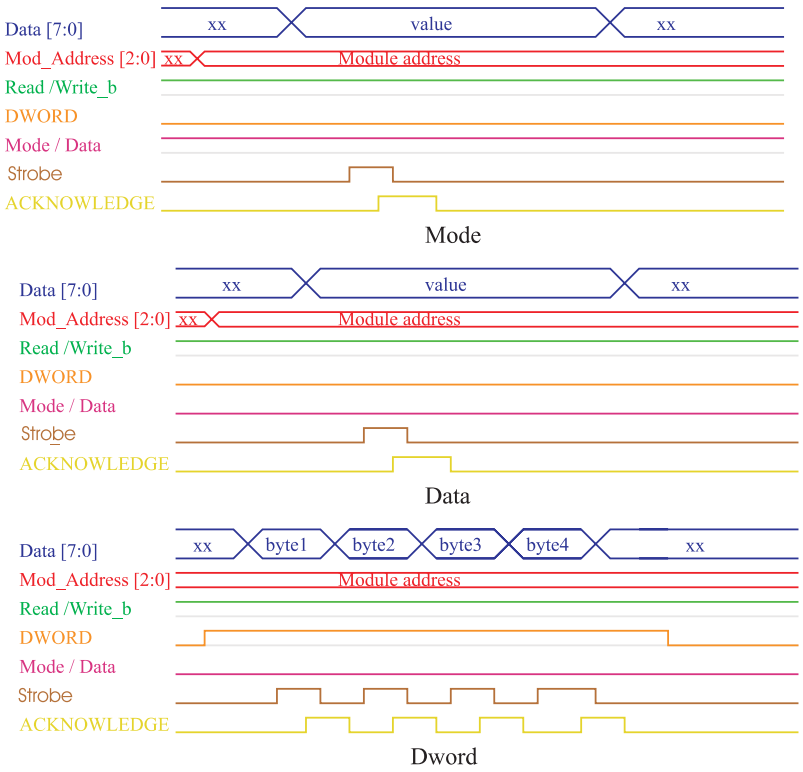


Figure 9: Signal timing on the BB (read accesses). Timing is not to scale. The WR_ CLK is a 16 MHz clock.

reads the implementation ID of module 2, you are in big trouble.
The BB has to be terminated. The terminations have to sit on the "last module", the module that sits closest to the end of the BB. The BB must only be terminated once (more than one termination resistor array will result in data transmission errors on the bus). As will be explained in the section about the TLU, the TLU was chosen to be the module that carries the termination resistors. Therefore it was given BB address 7, but this is just a convention. The BB address has nothing to do with the order of modules on the bus and can be chosen absolutely arbitrarily as long there is no software constraint on the BB address[11].

3.2 BB Interrupts.

The BB PCI card is capable of sending IRQs directly to the PCI bus and to the CPU. There are various reasons why such an IRQ can be generated by

[11] In fact, the software at the moment only enforces BB address 7 for the TLU.

Code	Access	Description
*mode = value	Write mode	*Value* is stored in the modules MA register. *Value* has to be an 8 bit type.
value = *mode	Read mode	Content of modules MA register is stored in *value*. *Value* has to be an 8 bit type.
*data = value	Write data	*Value* is stored in the register currently selected in the module. *Value* has to be an 8 bit type.
value = *mode	Read data	Content of the register selected in the addressed module is stored in *value*. *Value* has to be an 8 bit type.
*dword = value	Write DWORD	*Value* is stored in the register currently selected in the module. *Value* has to be an 32 bit type.
value = *dword	Read DWORD	Content of the register selected in the addressed module is stored in *value*. *Value* has to be an 32 bit type.

Table 2: Overview about BB accesses. At system startup, the pointers have to be initialized properly by the software for each BB module.

the BB PCI card. The most important thing here is, that the BB FPGA can generate an IRQ which will be sent to the PCI bus and the CPU, if a certain line from the BB itself, the IRQ line, becomes active. This is the so-called BB IRQ, which can be caused by any of the modules connected. This IRQ line is shared, i.e. more than one (all, in fact) can drive on this line at the same time. The conditions which have to be met to make a module pull the IRQ line active depend on the implementation of the module's firmware and the current module settings.
The BB FPGA contains, just like the ISRs in the different telescope modules, an 8 bit interrupt status register. In contrast to the modules ISRs, which support up to 4 different IRQ addresses, the BB FPGA supports only 3 different IRQ addresses. The fourth bit is a global enable for all IRQs which can be set by the BB FPGA. These are:

- *XINT* (ISR bit 0). This bit is set if any of the other three bits is set for what reason ever. Indicates a kind of global BB FPGA IRQ.

- *XINT TIMEOUT* (ISR bit 1). This bit is set, if the strobe of a read or write access to a certain module is not answered properly with an acknowledge signal.
- *XINT ERROR* (ISR bit 2). This bit is set as a result of an internal error in the state machine generating BB accesses. Please refer to the manufacturer's documentation about the BB.
- *XINT MODULE* (ISR bit 3). This bit is set if the BB IRQ line is pulled active by one or more modules connected to the BB. This is what in the following will be referred to as BB IRQ, the interrupt on the Blueboard bus.

The enables for these IRQs are the corresponding MSBs, so bit 4 is the global enable, bit 5 is the timeout IRQ enable etc.. To enable an IRQ, one not only has to enable the corresponding ISR bit but also the global enable. Also, the AMCC IRQ has to be enabled to allow the AMCC to pass the BB FPGA IRQ to the PCI bus. But normally this is done automatically by the startup software.
As described, during startup the software performs a scan over the entire BB address range to "detect" connected modules. This is done by disabling the timeout IRQ, sending a read access to MA 0 and, after some time, reading back the BB FPGA ISR. If the timeout IRQ is detected there, there is no module connected at this address, if not, a module is connected and the result of the read address can be taken to determine the module type. The IRQ is disabled to prevent the software from reacting to this artificial timeout like on a real timeout during system operation. For system operation, of course the timeout IRQ is enabled again. During system operation, a timeout IRQ indicates something going really wrong. The possible causes for this condition normally are either missing BB cables, missing supply voltages (not necessarily for the module which has been accessed) and so on.
If the CPU detects an interrupt request on a certain interrupt, all modules sharing this interrupt are asked if it was them who generated this IRQ. If the BB PCI card is asked, the AMCC will know that the IRQ was generated by the BB FPGA and a "Handle interrupt" function will automatically be called, but to find out what the reason of the IRQ was, one has to read the BB FPGA ISR. As it is for all the modules, the ISR of the BB PCI interface can be addressed by reading MA 1 (at BB address 0, of course). Comparing the LSBs of the ISR with its MSBs one can easily find out what the reason for the IRQ has been. If, and only if, a BB IRQ has been the reason, the IRQ handler will access the ISRs of all connected modules to find out which

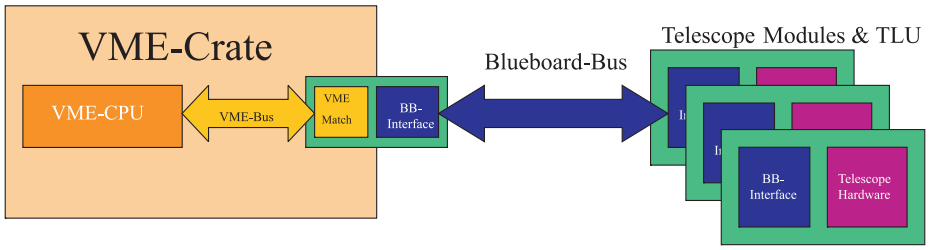


Figure 10: Setup for VME based operation of the BB and the BAT. The role of the BB busmaster is taken by the BB VME interface, which, on its part, is controlled by the VME busmaster. This VME busmaster can be an common PC VME interface or an embedded PC / VME CPU.

module sent the BB IRQ (and, of course, why). So this IRQ processing loop is, where all the changes for different DAQ modes have to be made.
The loop will, after having found and processed one module's IRQ, not return to an idle state but continue its loop through all modules connected, so all pending IRQs will be processed at a time and the IRQ source does not have to be determined more often than necessary.

3.3 The BB VME interface

As described, VME components for DUT readout can easily be taken into the readout via a PC-VME interface. DUTs can also be integrated into the BB environment by using the DTC card as an interface to the BB. There is a third way of bringing the VME-DUT world and the BB world together: The BB can be operated not using the PC as platform, but a VME-CPU, and the BB busmaster is not the BB PCI interface but a BB VME interface, whose timing and behaviour on BB side do not differ from the BB PCI interface's behaviour. On VME side, it is connected to the VME bus via a standard VME access interface circuit and can be controlled by the VME busmaster, which then is a VME CPU, an embedded PC or another PC-VME interface. A schematic overview of this setup is shown in fig. 10. As this is the solution which will be used to operate the BAT in the future by the ATLAS collaboration, such an interface is currently under development. Some re-programming will have to be done, especially concerning the IRQ handling and the software's class structure, and neither OS nor programming language are defined yet, and it is difficult to make a speed forecast, as the whole system was designed for PC based operation.

4 Module hardware

4.1 The modules' case

A module's case is what is visible from the outside. If the module works correctly and is properly set up, a user won't ever have to open the case and see what is inside. The case is meant to combine mechanical protection, handling fixtures and fittings, alignment marks and plugs and connectors for connecting peripherals. It has to be easy to assemble, parts have to be interchangeable, exchange should be easy and, of course, some very special demands arise from the fact that the device will be used in a very special environment. The modules consist of a solid aluminum frame, in which the different PCBs are mounted. A schematic is shown in fig. 11. The circuits will be described in detail in separate sections. A drawing of the frame (without cover plates) is shown in fig. 12. There are two compartments inside the case. The (smaller) front compartment[12] holds the hybrid assembly, the (larger) rear compartment[13] holds the ATC and the DTC PCBs. All connections to the outside needed by the PCBs are fed through fittings on the rear side of the case. Fig. 13, a schematic rear view (not to scale) of the modules rear side, shows the positions and purposes of the different connectors. Fig. 14 shows a photography of the same section. Cables from the power supplies, the TLU and the BB are connected to these plugs. The two 50 pol. IEEE 488 connectors are in- and output of the BB cable. It does not matter which BB cable is connected to which BB port. The large 8 pol. LEMO connector is for connecting the low voltage module power supplies, the small 2 pol. LEMO is for connecting the detector bias. The three single port LEMO connectors serve as in- or output for status signals. Inside the case, the BB on the DTC and the power cables on the ATC side are also connected to their PCBs using plugs. The LEMO connectors, the LED and the DTC power connections are soldered on both case and PCB side. For the newer modules the LEMO plugs are connected to the DTC using grommets. To ease assembly and to avoid wrong connections, all connectors enforce the right polarities. It is planned to replace the remaining solder connections by plugs to further increase modularity and ability to exchange parts. The four LEDs are connected with the corresponding LED on the DTC PCB. Please note that for the four modules of the prototype series sometimes the order of LED got messed up[14], so it might vary from module to module. Although a

[12] In the following, this will be referred to as detector compartment (DETC).
[13] This will be the electronics compartment (ELEC).
[14] This is the case especially for module 2.

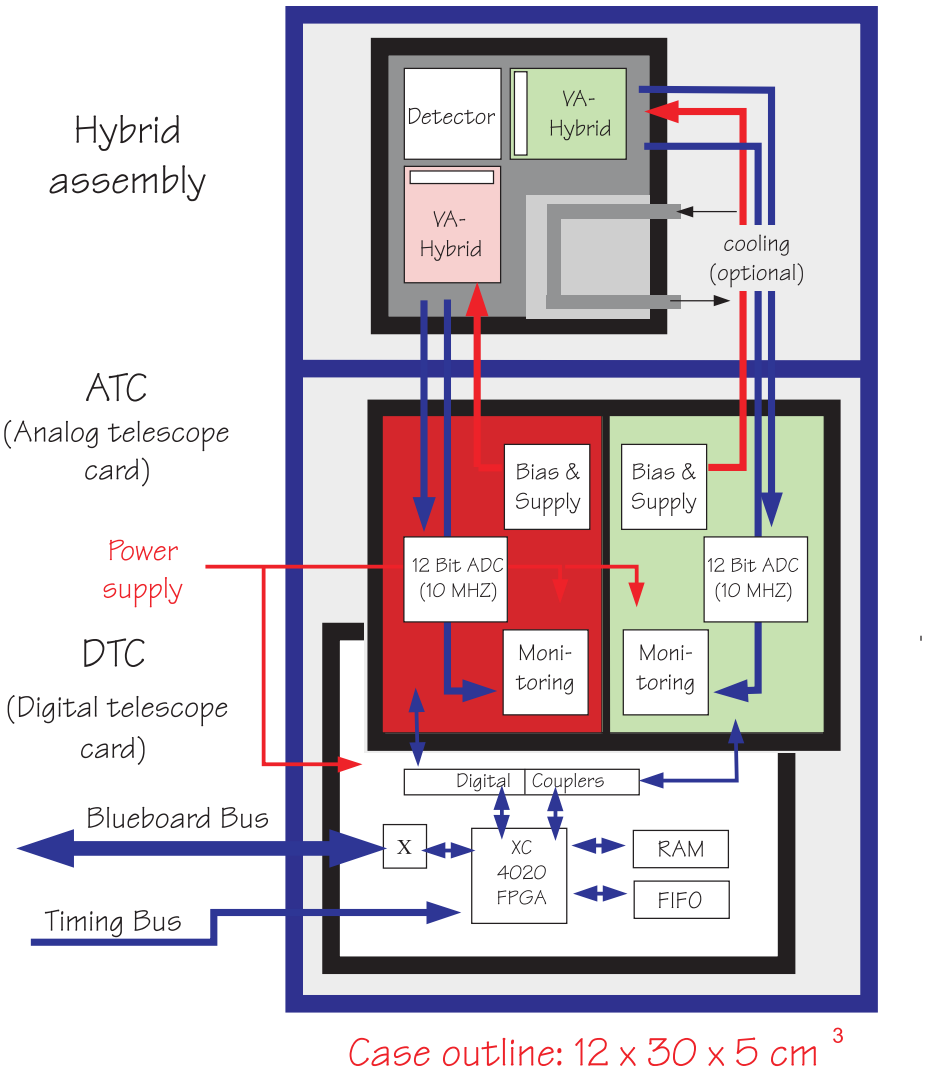


Figure 11: Schematic of a BAT module setup. The two case compartments contain the hybrid assembly carrying the FE electronics and the detector and the full data acquisition electronics.

red LED in the case always corresponds to a red LED on the DTC, they are not always fixed to the same location on different modules. But the user can easily determine their function from their behaviour after reading the DTC section of this manual.

The PCBs are mounted in the ELEC as a PCB sandwich connecting the two PCBs back-to-back. The ELEC can be closed with two aluminum cover plates, one on the rear and one on the front side of the module. The cover plates provide as well mechanical and electrical protection (especially as far

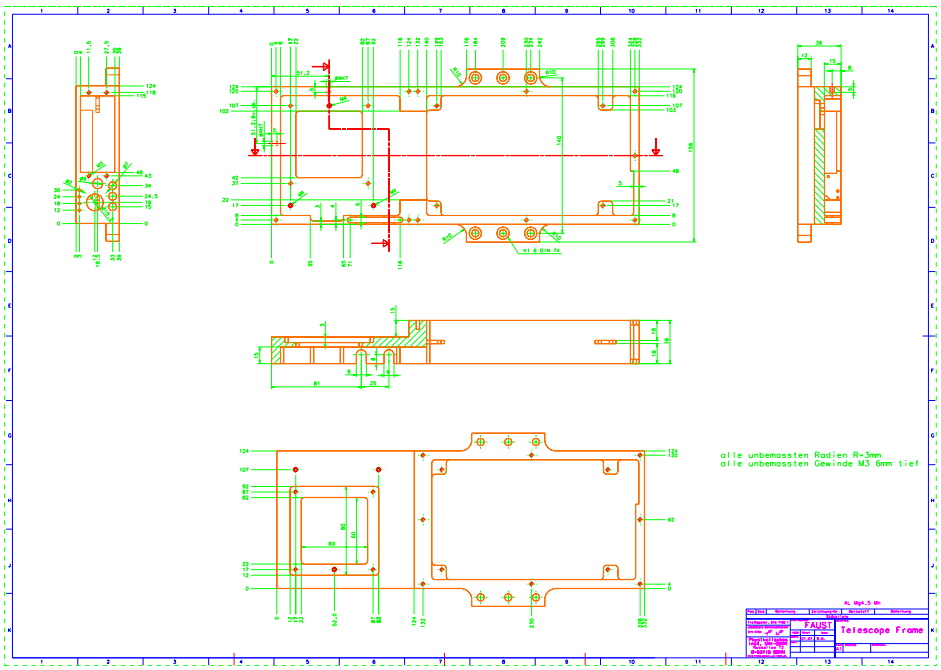


Figure 12: Drawings of the module's frame. The layout of the cover plates is not shown. The missing covers exist of 4 parts: 2 larger aluminum plates, which serve as cover for ELEC, and a small frame and another plate, which serve as DETC cover.

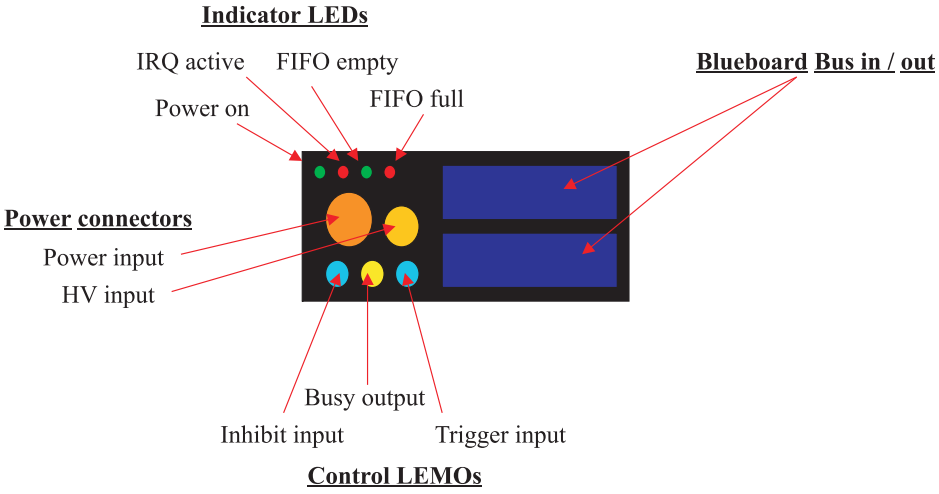


Figure 13: Schematic back side view of a module showing the positions of the different connectors and indicator LEDs. For the modules of the prototype series, the positions of the latter may vary.

as the high voltage section on the ATC is concerned) as electromagnetic shielding of the components (especially important for the DETC), so care has to be taken to provide a conductive connection between the case and the cover plates (especially in case the frame and cover plates are anodized).
The hybrid assembly is not directly fixed to the case. Between the screw-holes in the case and the hybrid assembly three 2 mm thick washers have to be placed which decrease thermal conductivity between the hybrid assembly and the case. This is because the two ELEC PCBs generate much more heat that the hybrid assembly does; thus the ELEC is prevented from heating the hybrid assembly. It is possible to cool the hybrid assembly using an external chiller. The assembly frame for the hybrid assemblies has a pipe structure implemented for this purpose and grommets for cooling pipes have been implemented in the module's case. Small aluminum pipes were manufactured which can be attached directly to the hybrid assembly and were supposed to serve as in- and outflow for coolant of any kind, but the possibilities offered by this feature are not yet fully exploited.

Figure 14: Photography of a module's back side holding the connectors. The LED are hardly visible, because they are not lit. The blue IEEE 488 connectors have been replaced later by a more reliable type which is black.

The cover for the DETC consist of two parts: The back side of the DETC is mostly covered by the case itself, leaving only a small window directly under the detector as pass-trough region for the beam. As this window has to be covered with as little material as possible to reduce radiation length, a frame has been manufactured which fits into a flange on the back side of the case and can be filled with material that has the desired properties, namely low radiation length, mechanical solidity and electrical shielding[15]. The best results were achieved with either two layers of aluminum foil or a single layer of copper plated capton. Care has to be taken here as well to provide a good electrical contact between shielding and case. The front side of the DETC is covered with an aluminum cover plate with a window left out where the detector is. This window is also to be covered with the same shielding material as is used on the back side. Also, some sealing has to be attached at the edge of the cover plate which covers the spacing between DETC and ELEC. This sealing has to be light tight and enables, if attached, the user to probe on the ATC with depleted detector and the ELEC cover on ATC side removed. The sealing reduces contributions to the detector leakage current generated by incident light and preserves normal system operation while debugging. Also it provides some protection for the DETC against dust and dirt if the ELEC cover is removed. However, care has to be taken to open the cover of the DETC only with the detector bias voltage off and when it is not avoidable. It has to be done under conditions where it can be made sure the hybrid assembly is not damaged and where it can't become dirty. Also, some precautions have to be taken while debugging on the ATC of a depleted detector module even when the DETC is closed. These will be explained in the section about the ATC. A picture of a fully covered module is shown in fig. 15. The origin of the telescope coordinates is given by the first channels on both P- and N-side hybrids. The position of this origin is also shown in fig. 15. The case also provides some fittings for the connection of the module to some support structure in the testbeam area. In fig. 15 one of the fittings is visible as the "wing" on the long side of the module. The positions of the modules relative to each other and relative to the beam and the coincidence counters depend as well on the beam profile shape and direction as on the beam parameters like beam energy. To minimize the influence of coulomb multiple scattering caused by the material unavoidably present in the beam line to the trace recognition accuracy especially at low beam energies it might be necessary to put two modules' detectors as close to each other as possible. This was done at the E3 testbeam facility in Bonn. The shape of the case was optimized for this kind of arrangement. The setup is shown in fig. 16.

[15]The cover also has to be opaque.

Figure 15: Photography of a fully covered module (front side). The ELEC and DETC covers can clearly be distinguished. The beam window is covered with copper plated capton. The origin of the module coordinates (crossing of strip 1 on N side and strip 1 on P side) is indicated by an arrow.

To ease handling and to as well set up as determine the alignment of the

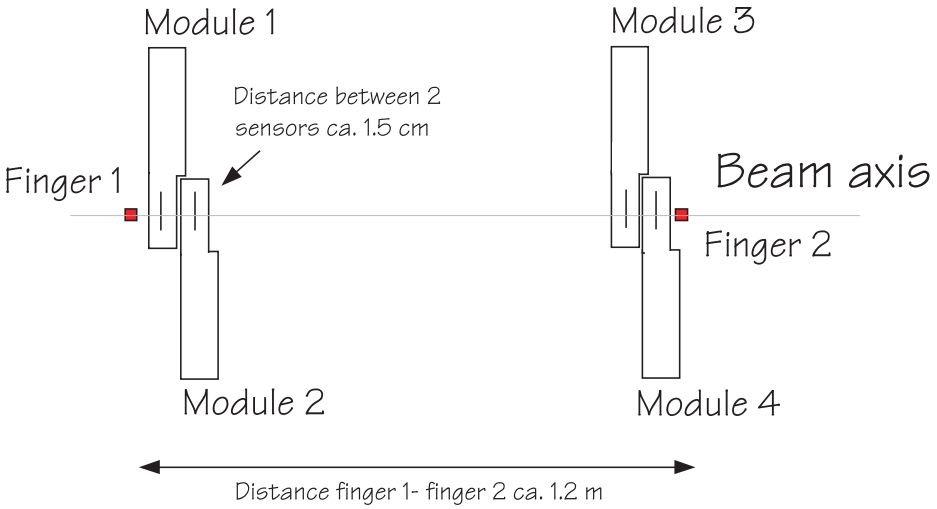


Figure 16: Testbeam setup at E3. The special arrangement of the modules is used to minimize influence of coulomb multiple scattering The shape of the modules helps to bring two modules sensors as close together as possible.

modules relative to each other and to the beam, some alignment marks at the outside of the DETC mark the position of the detector center. There are two alignment marks for each module. They are made up of two cylindrical holes in the frame. Their position on the case is at the point where the center of the sensor inside the case is meant to be for the corresponding dimension. So the position of the sensor center is at the cross-over point of the lines

parallel to and through the middle of the alignment holes[16]. It is planned to manufacture small cones which fit into the holes and have their apex right above the hole center, so alignment will be even more easy.

4.2 The hybrid assembly

The hybrid assembly is the most advanced mechanical part of the module. It carries as well the front end electronics for detector readout as the detector itself. It has to provide mechanical support for these parts and easy handling possibilities. Once a hybrid assembly is assembled, it forms a unit together with the two hybrids and the detector it is assembled with. It is not possible to disassemble an assembled hybrid assembly without at least destroying the detector. The mechanical structure is quite complex. For more details please refer to the drawings of the hybrid assembly parts. It has to bear the large area silicon strip detector, only being allowed to support it under the very edges and in no case directly under the sensitive area. To simplify bonding and to avoid putting unnecessary material into the beam line, the hybrids are mounted right angled[17] to each other. So no Z-print is needed at the expense of a somewhat more complicated structure of the assembly frame. The assembly frame, which is the "solid" part of the hybrid assembly, is manufactured from a monolithic aluminum block. It is designed to combine mechanical reliability, low thermal expansion and good cooling efficiency with good accessibility to ease bonding. As mentioned, the hybrid assembly can be cooled making use of a pipe structure in the assembly frame. It has to be manufactured very precisely. As the silicon detector is very thin, small mechanical tensions can be hazardous and may result in damaging or even breaking the detector. For the same reason, gluing has to be done very carefully as well.
A schematic view of the hybrid assembly is given in fig. 17. An overview about the assembly steps is shown in fig. 18. The front end hybrids are not directly glued to the assembly frame. For each hybrid a hybrid carrier structure is manufactured. As the hybrids consist of a PCB glued to an aluminum oxide ceramic carrier, the hybrid carrier also consists of aluminum oxide ceramic plate, the backplane, with some drill holes used to fix the carrier to the assembly frame. 2 ceramic strips are glued to the backplane in a T-shape, thus forming either a pocket, in which the hybrid itself is fixed, and a "L" structure, which will form the support for the detector's edge on

[16] Of course, due to assembling inaccuracies the real position might be somewhat different from the ideal position, so the actual alignment still has to be calculated from the data.
[17] The stereo angle of the detector is 90°.

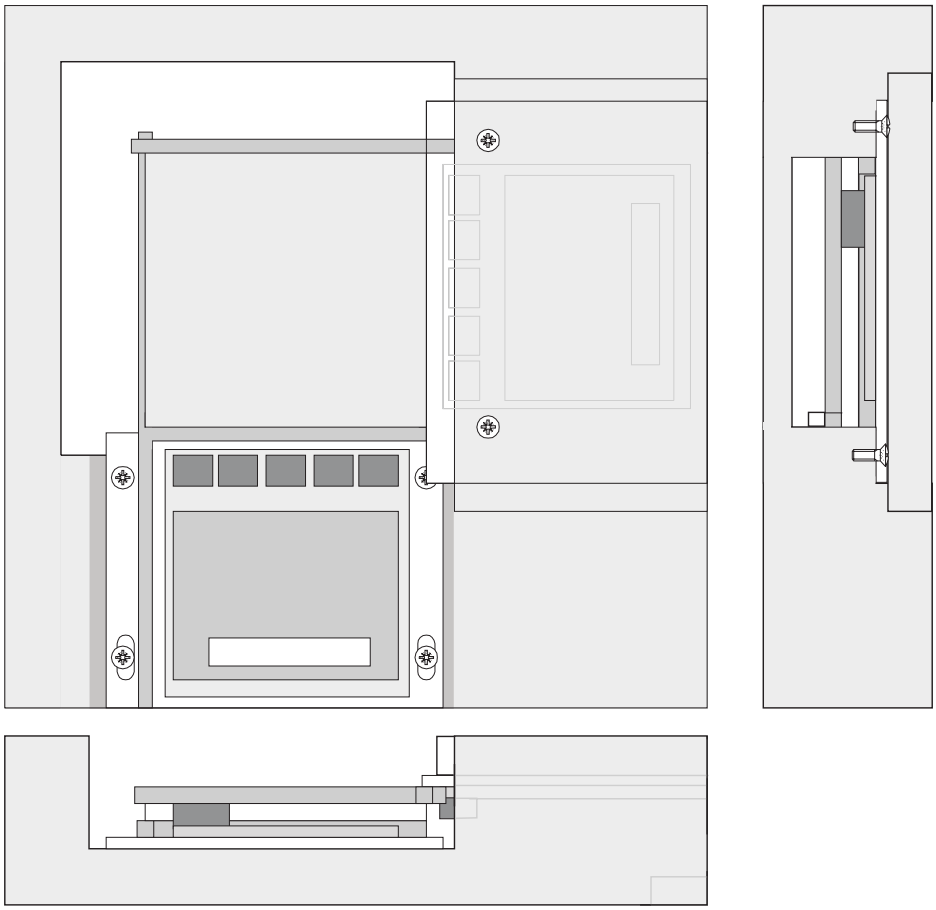


Figure 17: Schematic drawing of a fully mounted hybrid assembly. Different colours indicate different parts. The pipe structure for coolant is not visible. In the projections of the assembly to the small sides, only one FE IC is shown for simplicity. For this arrangement of the hybrids the "origin" of the telescope, being the crossing of the strip centers for the strips 1 on both detector sides, is in the lower right corner of the detector.

two sides. As this gluing process has to be very precise and reproducable, a special gluing tool is used to make the parts sit at the exact position with high accuracy. The hybrid is glued to the hybrid carrier's backplane using an elastic silicon adhesive which can compensate thermal expansion. The fully assembled hybrid carrier is then fixed to the assembly frame. Four screws, which are only slightly tightened, determine its position on the assembly frame. The screw holes in the ceramic plate are quite large, so there is some fetch, but the hybrid carrier's back side is covered with thermal grease, which as well serves as an adhesive as it manages to compensate differences in the thermal expansion coefficient between the ceramic hybrid carrier and

the aluminum of the assembly frame. In the next step, the sensor is glued to the L-shaped ceramic of the first hybrid carrier, and the second hybrid carrier is then inserted in its fittings on the assembly frame head first and right-angled to the first hybrid carrier and fixed with screws. The detector is glued to this hybrid carrier's "L" structure as well. Care has to be taken here, for due to the shape of the assembly frame the second hybrid won't be accessible any more after gluing (except for the FE-ICs bond pads and the connector to the ATC, of course). So, all modifications necessary for this hybrid have to be made before fixing it to the hybrid assembly. Please note that the hybrid on the first hybrid carrier should always contact the p-side of the detector[18], so be careful. To make a connection between the second hybrid carrier (the one that contacts the detector n-side) and the assembly frame with good thermal conductivity, the back side of the carrier is covered with an aluminum cooling plate which is screwed to the assembly frame. The spacing between hybrid carrier ceramics and aluminum cooling plate is filled with thermal grease. The cooling plate also provides mechanical protection for the second hybrid carrier's backplane. Now the hybrid assembly is ready for bonding.

If the hybrid assembly is set up correctly, the detector surface should be on a level with the surface of the FE-ICs and the distance between the bond pads should be as small as possible on both sides. So bonding the hybrid assembly should be easy. Whenever dealing with the hybrid assembly, one has to be extremely careful not to damage the bonds or to mechanically stress the detector itself, for it is only fixated with small thin strips of "brittle" ceramics.

After assembling the hybrid assemblies of the prototype series it turned out that the common mode noise can be significantly reduced by adding a large additional filtering low-pass circuit for the detector bias voltage directly to the detector. The low pass capacitor had to be placed as close as possible to the hybrids, so it had to be glued directly to the assembly frame itself. A view of a fully equipped module's DETC is given in fig. 19. The cables used to connect the hybrids with the corresponding ATC compartment use small, expensive connectors manufactured by Omnetics on hybrid side. On ATC side, standard IDC connectors were used. When manufacturing the prototype series modules, it was not possible to provide the Omnetics type connectors with flat ribbon cable needed to make good contact to the IDC connectors, but with single wire contacts only. It was difficult get a reliable contact between hybrid and ATC that way. This will hopefully be fixed in later versions, either by using another type of connector on the hybrids

[18]There is, of course, no physical reason for this. It's pure convention.

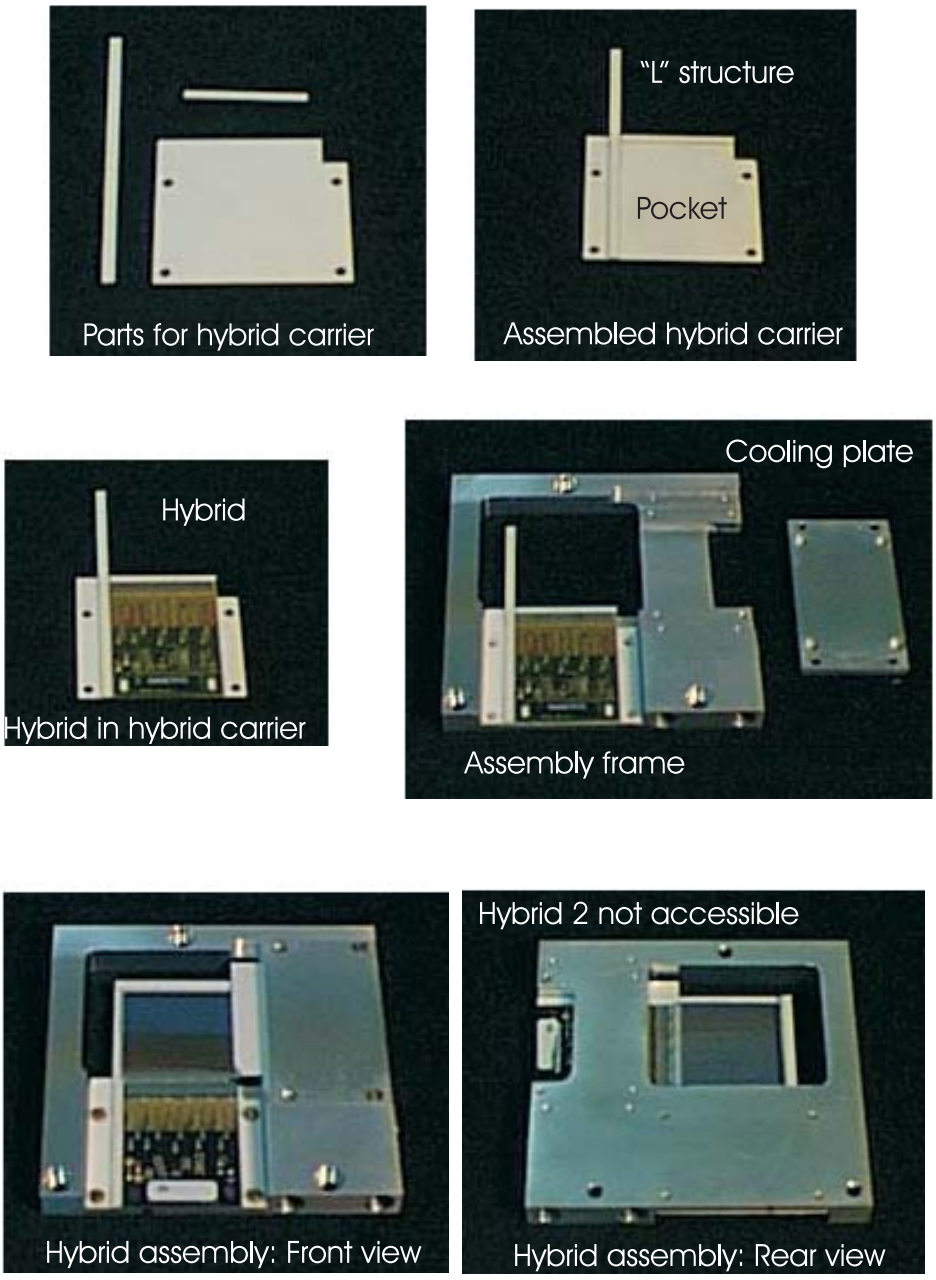


Figure 18: How to assemble a hybrid assembly. First line: Parts, of which a hybrid carrier consists and fully assembled hybrid carrier. Second line: Hybrid attached to hybrid carrier and inserted into the assembly frame. Detector and second hybrid are attached simultaneously in the next step. Third line: Fully assembled hybrid assembly viewed from both sides.

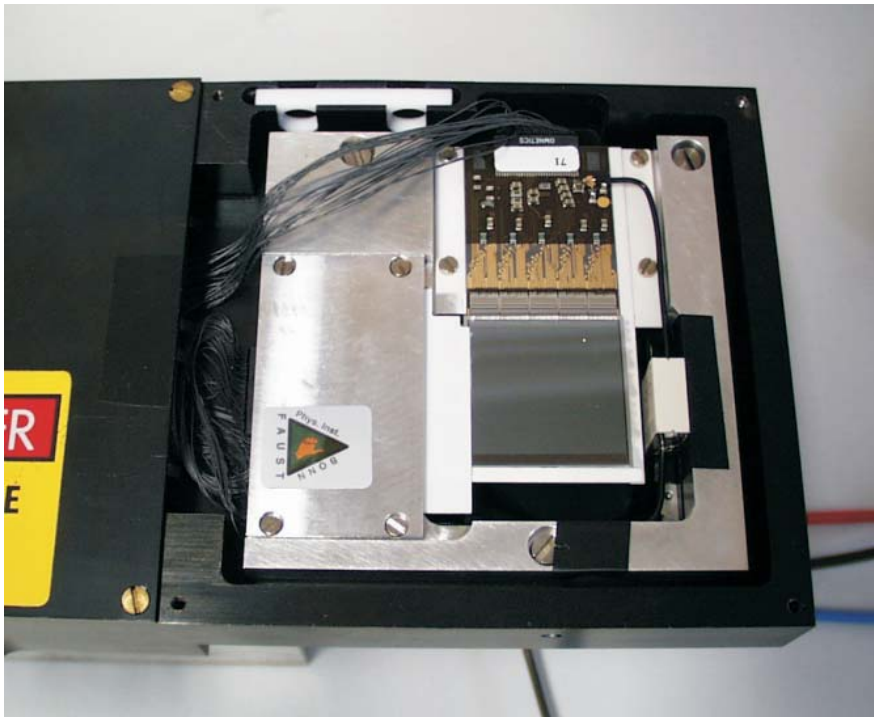


Figure 19: A view of a module's DETC equipped with a hybrid assembly. The hybrid assembly has an additional low pass capacitor at the right edge of the frame. The interconnection cables from the p-side hybrid to the ATC are also visible.

replacing the expensive Omnetics type or by using flat ribbon cable instead of single wires for the connection to the IDC plug.

Gain	25 μA / fC
ENC	80 $\pm$ 15 e^- / pF @ 1 μs shaping 60 $\pm$ 11 e^- / pF @ 2 μs shaping
Dynamic range	$\pm$ 4 MIPs
Peaking time	0.5 - 3 μs
Power consumption	1.2 mW / channel
Applications	Medium sized detectors with $\approx$ 10 pF load capacitance

Table 3: Overview about the parameters of the VA 2 IC. ENC values are given depending on the load capacitance on the VA input.

5 Front end electronics

Only a brief description of the BAT front end electronics is given here, as a much more detailed description can be found in the publications listed below. Concerning the VA 2 ICs:

- *The VA 2.* Specifications & manual. Version 1.4. Published by IDE on May, 14th, 1997.

Concerning the VA hybrids:

- Technical design report for BELLE readout hybrid. Author: Bjørn Magne Sundal. Published by IDE on October, 7th, 1997.
- BELLE SVD Hybrid, Conceptual design report. Version 2.3, published by IDE on August, 13th, 1997.

5.1 The VA 2 IC

The VA 2 IC is a 128 channel low noise and low power charge sensitive preamplifier-shaper circuit with simultaneous sample and hold, serial analog readout and calibration facilities. A schematic of one channel and the chip's read out serializer circuit is shown in figure 20. An overview about the most important parameters of the VA 2 is given in table 3. The IC is controlled via 5 digital input signals:

1. *Hold:* (Active low) The hold signal toggles between sample and hold mode of the VA 2 circuit. With *hold* being inactive, the voltage signal on every channel's sampling capacitor follows the preamplifier/shaper

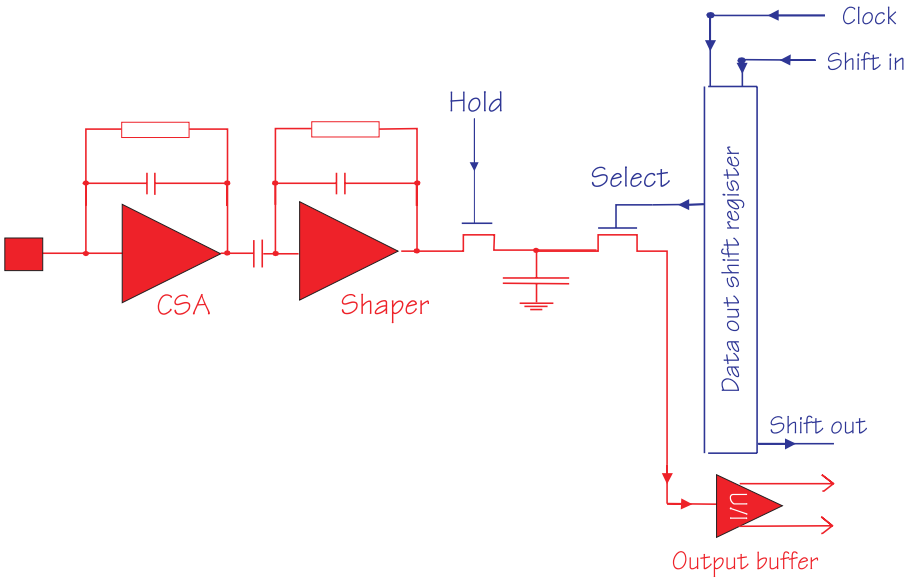


Figure 20: Simplified schematic of a single VA 2 channel and the chip serializer structure. The *hold* signal toggles between sample and hold mode. The *clock* can be operated up to a frequency of 10 MHz. Via *shift in* data can be fed into the VAs serializer shift register. The channels selected in this way put the voltage signal on their sampling capacitor to the outside using the differential analog output buffer.

output. If *hold* becomes active, the sampling capacitor is undocked from the preamplifier output and the voltage level is stored.

2. *Shift in:* (Active low) This is the input line for the IC's serializer shift register and for the IC's calibration shift register.

3. *Clock:* (Active low) This is the clock signal for both, the IC's serializer and calibration shift register.

4. *Digital reset:* (Active high) With *digital reset* the contents of both calibration and serializer shift register are reset to zero (zero here means inactive).

5. *Test on:* (Active high) This signal toggles, whether the serializer shift register or both serializer and calibration shift register receive the *clock* signal.

The VA 2 is equipped with a RCCR shaper for each preamplifier cell, whose shaping time can be varied with some bias parameters. The bias parameters are listed in table 4. Typical values are 1-2 μs. The advantage of

Name	Task	Typical value
V_{fp}	Control voltage for feedback resistor in preamplifier	-200 mV
I_{pre}	Bias current for preamplifier	500 μA
V_{fs}	Control voltage for feedback resistor in shaper amplifier	700 mV
I_{sha}	Bias current for shaper amplifier	22 μA
I_{buf}	Bias current for differential analog current output buffer	140 μA

Table 4: Bias voltages and currents needed by the VA 2 IC

a RCCR shaper is a peaking time t_{pk} which is nearly independent from the total signal amplitude.
The serializer shift register contains 128 cells, one for each preamplifier channel, and one analog output buffer, whose output signal consists of a differential analog signal current. Whenever the shift register cell for a channel is active, the sampling capacitor for this channel is connected with the analog output buffer's input. The (differential) current flow out of the analog output buffer is given then by

$$I_+ = \frac{I_{buf}}{2} + \frac{I_{signal}}{2}$$
$$I_- = \frac{I_{buf}}{2} - \frac{I_{signal}}{2}$$

with I_{buf} being the buffer bias current, one of the VAs bias parameters, and I_{signal} the signal voltage on the sampling capacitor converted to a signal current. This signal consists of a channel specific pedestal value, which is simply the default output voltage of the preamplifier channel, and an "event part", which is generated by additional charge generated by an event on the input of the preamplifier channel.

$$I_{signal} = I_{pedestal} + I_{event}$$

For the event part of the signal current, typical gain values are 25 μA signal current per fC injected charge. Problems arise if there is more than one shift register cell active. As it is impossible to separate the signals, care has to be taken to make sure that there is only one shift register cell activated at a time.
In the following, a typical readout sequence is described as it is implemented in the BAT DTC firmware. The chip is not self-triggering, so an external trigger signal is needed to activate the (external) readout sequencer. In normal (*test on* is inactive) operation mode, this external trigger signal indicates

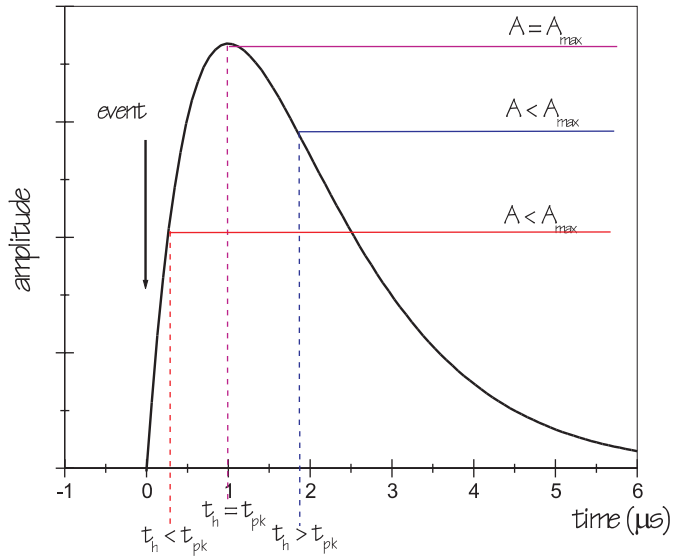


Figure 21: Hold timing of the VA 2 IC. This is the ideal situation with t_{tl} = 0 (no trigger latecy). The peaking time can be adjusted with some bias parameters of the IC. Typical values are 1-2 μs. Choosing hold times either too long or too short will yield in signal loss.

the occurrence of an event and start the readout sequencer. Depending on the trigger latency[19] t_{tl} the readout sequencer has to activate the *hold* signal together with the preamplifier signals reaching their maximum or at least as close to this point of time as possible. Ideally, the *hold* signal would be turned on when the hold time t_h, which is given by

$$t_h = t_{pk} - t_{tl}$$

has passed. Otherwise, the bad timing results in signal loss (see figure 21). To prevent unwanted data in the output serializer shift register, the sequencer gives a *digital reset*, and the serializer shift register is initialized with a single "one" and one clock pulse at the beginning. With another clock pulse, the next cell is selected and so on. The output signal then is a series of 128 differential analog signal currents. During the readout of a singe VA, the hold signal has to be kept active. The 128th shift register cell's output is directly connected to the *shift out* output pad. If all went well, a single *one* appears at this output after the 128 th clock pulse, indicating that now the 128th channel is active and thus the readout of the IC has been successfully completed. The hold signal can be released afterwards. A typical readout

[19]The delay between the actual event and the arrival of the trigger signal generated by some trigger detectors.

sequence is displayed in fig. 22.
The noise behaviour and signal response of a single preamplifier channel can

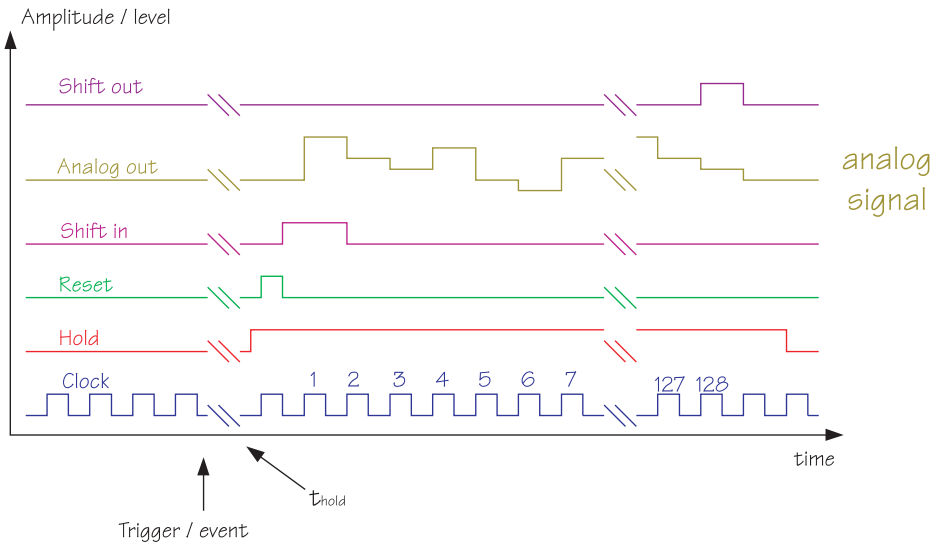


Figure 22: Readout sequence of a VA 2 IC. Only logic levels are shown independent from the "real" signal polarity.

be observed by activating the corresponding serializer shift register cell, thus putting the voltage signal of the sampling capacitor on the output buffer, and disabling the hold signal.
When the IC is operated in *test mode*, the *calibration register*, a second 128 cell shift register running in parallel to the serializer shift register, is initialized and clocked together with the serializer shift register. If a preamplifier cell is selected with the *calibration register* an analog input of the VA 2, the *cal* line, is connected directly with the preamplifier input. Injecting small charge signals on the *cal* line will yield a pulse answer, which can be observed directly at the analog output as described above. This is a common procedure to calibrate the ICs and to adjust bias parameters.
The VA 2 IC is designed for medium preamplifier load capacitances. The bond pads for the preamplifier channel inputs are arranged in two staggered rows with 50 μm pitch. For each channel, an additional "spare" bond pad is present.

5.2 The BELLE hybrid

The VA 2 ICs are suitable for *daisy chained* readout. The *shift out* signal of a VA can serve as *shift in* signal for another VA receiving the same clock signal.

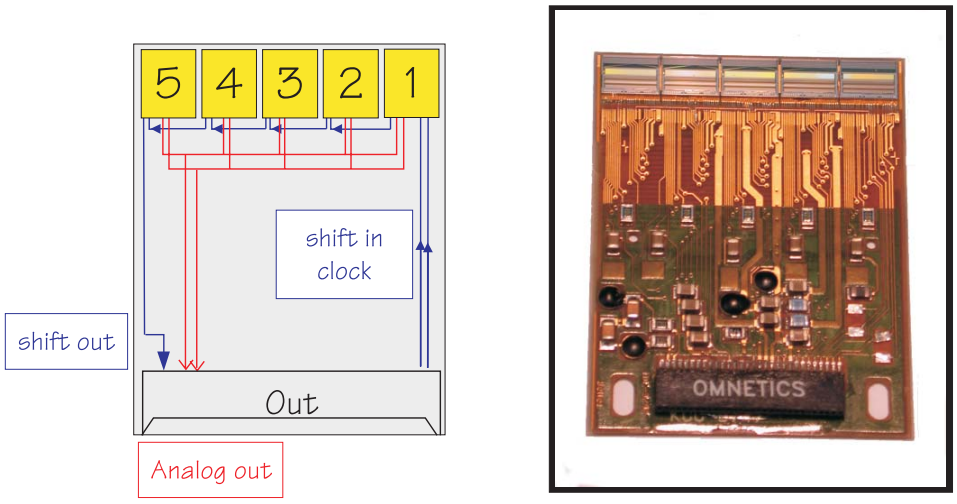


Figure 23: Picture and schematic view of a BELLE VA 2 hybrid. Several passive components provide decoupling of bias and supply lines. The first VA being the VA which is initialized first, is the rightmost one.

Thus, the 2 × 128 cell shift registers of the 2 VA behave like a 256 cell shift register of a larger VA. If one connects the differential current output signals of the two VAs directly together[20], the differential current signal observed on this line also behaves like the output signal of a 256 channel VA. Of course, not only the clocks of the 2 VAs are connected in daisy chained operation. All digital control signals are connected together with the *shift out - shift in* connection being the sole exception.
This feature is especially useful, if a detector larger than 128 × 50 μm is to be read out. Operating two or more VAs daisy chained enables the user to read out the whole detector at once using only one sequencer and signal digitization circuit at the cost of a somewhat slower serial readout. Supply and bias voltages have to be adapted to the new higher current draw, and, as far as bias parameters and shaping times are concerned, a compromise has to be found between effort and flexibility: the preamplifier gains and, even more important, the shaping times should be adjustable to the same value for all the different VAs.

In the BAT setup, detectors of 3.2 × 3.2 cm^2 and 50 μm strip pitch were used, so 640 VA channels corresponding to exactly 5 ICs were needed. This was the reason why it was decided to use the BELLE SVD hybrids in the BAT setup. The BELLE hybrids were originally developed for the silicon vertex detector of the BELLE experiment at KEK in Japan. A BELLE

[20]This would not be possible for differential voltage outputs.

hybrid carries 5 daisy chained VA ICs on a ceramic carrier together with a small 2 layer PCB providing all control signals, bias and supply voltages to the ICs. The PCB also holds some resistors and capacitors for power line and bias decoupling. All signals are fed to the hybrids by a single connector. As described, the readout is daisy chained and serial. The ceramic carrier provides good thermal conductivity to the support structure. Moreover, the small difference in the thermal expansion coefficient minimizes the stress on both sensor and FE ICs[21]. A photography of a BELLE hybrid together with a simplified schematic of the readout is shown in fig. 23. The full schematic of the hybrid is shown in figure24. All control signals except for em shift in are fed to all ICs in parallel, their output signal lines are connected together. The supply voltages, preamplifier and shaper bias currents are fed to all ICs by a single connector line, so care has to be taken when designing the current sources as a current draw five times higher that the current draw of a single VA will occur on these lines. For the shaper bias current, additional in-line resistors can be used on the hybrid[22] to regulate the current draw to the individual VA ICs. The buffer bias current supplies the current for the VAs' analog output buffer. This current only flows if an output shift register is active. Only one VA shift register should be active at a time, so this current has to be provided only once (same current draw as for a single VA IC).
The preamplifier bias voltage is also set for all VAs in common, but the shaper bias voltage can be set for each VA IC individually. As the shaper bias voltage has a large influence on the peaking time, this feature can be used to adjust t_pk of the different VAs to the same value.
The hybrid carries an integrated calibration capacitor for each VA. Two copper layers in parallel, the space in between filled with some dielectric, form a capacitor with a capacitance of ≈ 0.5 fC. One side of a calibration capacitor is directly to a VA 2 calibration input, the other side is connected with the calibration capacitors of the other hybrids and a calibration signal coming from the connector. The calibration signal is terminated with a 50 Ω resistor, which has to taken into account when calculation the amount of charge generated by an applied voltage step at the calibration capacitor's input.
Although the digital signal inputs of the VA are single ended, 2 bond pads are provided on the IC for clock, hold and digital reset. One of the pads (the non-inverted signal pad, as the VA uses active low logic) is not connected to the IC's circuits. In spite of that signal lines exist on the hybrids for these "dummy" signals. By driving differential signals on these signal lines, digital

[21] See also the section about the hybrid assembly.
[22] This feature is not used for the BAT hybrids.

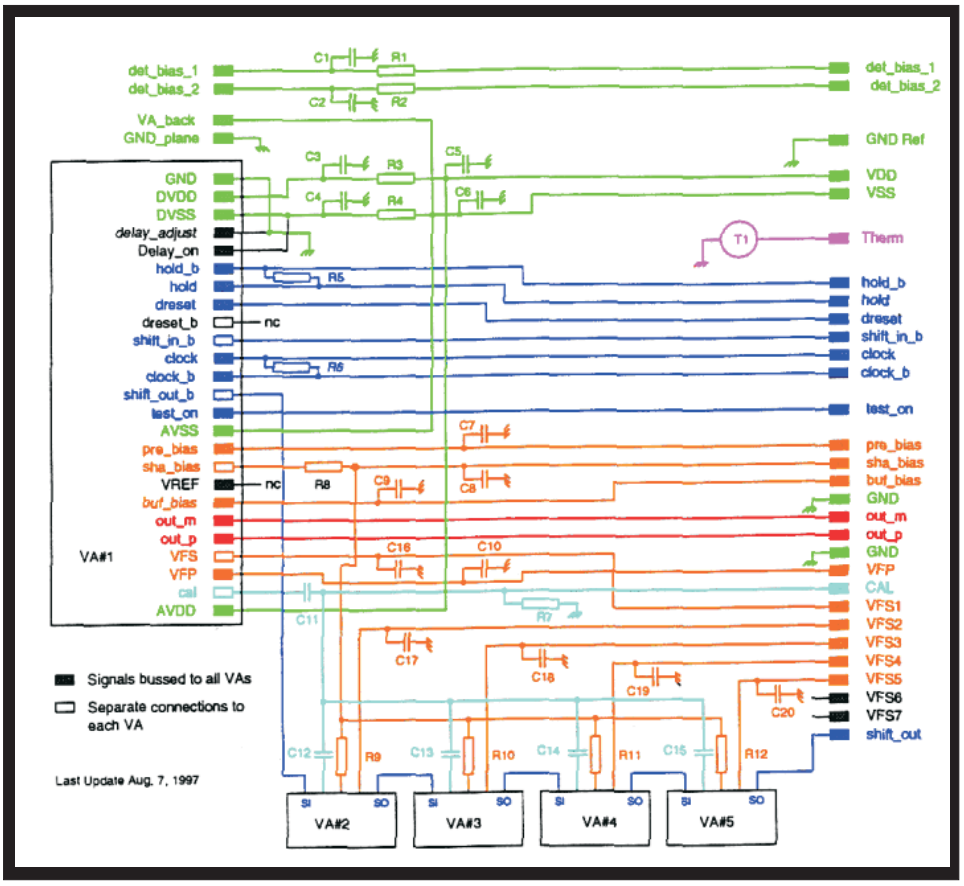


Figure 24: Full schematic of the VA hybrid. The distribution of digital and analog voltages and signals as well as passive components are shown. The detector bias circuit, however, had to be modified (see detector section). The different signal and power lines are colour coded: *Green:* Power supply lines. *Blue:* Digital signal lines. *Red:* Analog output signals. Orange: Bias voltages and currents. *Violet:* Thermistor.

to analog crosstalk, i.e. feed-through from digital signal to analog input or output lines, can be significantly reduced. This is important for those signals, which are to be switched while the ICs are in sampling mode (*hold*) or if the ICs are read out (*clock*). The dummy pad for digital reset is not connected is not connected on the hybrid. Between signal and dummy signal termination resistors were provided, which have been removed later due to problems with the Logic driver ICs. As the cable lengths are short, the missing termination is not critical.

Parameter	Value	Unit
Chip size	34.6 × 34.6	mm^2
Active area	3.2 × 3.2	mm^2
Thickness	300 ± 15	μm
Full depletion voltage (V_{fd})	80	V (max.)
Leakage current (at V_{fd})	1000	nA (max.)
Breakdown voltage	100	V (min.)
Number of strips	640	
Number of readout strips	640	
Strip pitch	50	μm
Strip width	20	μm
Readout AL width	16	μm
Bias method	AC coupling	
Bias resistance (polysilicon)	50	MΩ
Coupling capacitance (at 10 kHz)	70	pF (typ.)
Breakdown voltage of coupling capacitor	100	V (min.)
Load capacitance P-side at V_{fd}, 1MHz	6	pF (typ.)
Load capacitance N-side at V_{fd}, 1MHz	8	pF (typ.)
Passivation	SiO_2	

Table 5: Overview about the parameters of the S6934 type DSSD. The load capacitance for the detector N-side is larger than for the P-side due to the parasitic capacitances of the P-stop implantations. As VA IC's noise depends on the capacitive load, the detector N-side is expected to have higher noise. The coupling capacitance is frequency-dependent due to parasitic low-pass circuits formed by the differential coupling capacitance and the finite resistance of the aluminum strip readout contact. The value given for the leakage current is integrated over the whole detector surface.

6 The detector

As the detector type may vary for different telescope setups, only a brief description of the detector type used for the original Bonn ATLAS telescope setup is given.

6.1 Parameter overview

The detector used for the modules of the prototype series is a Hamamatsu product. It is a double sided silicon strip detector type S6934. It has a sensitive area of 3.2 × 3.2 cm^2 and 50 μm pitch. The pitch here equals

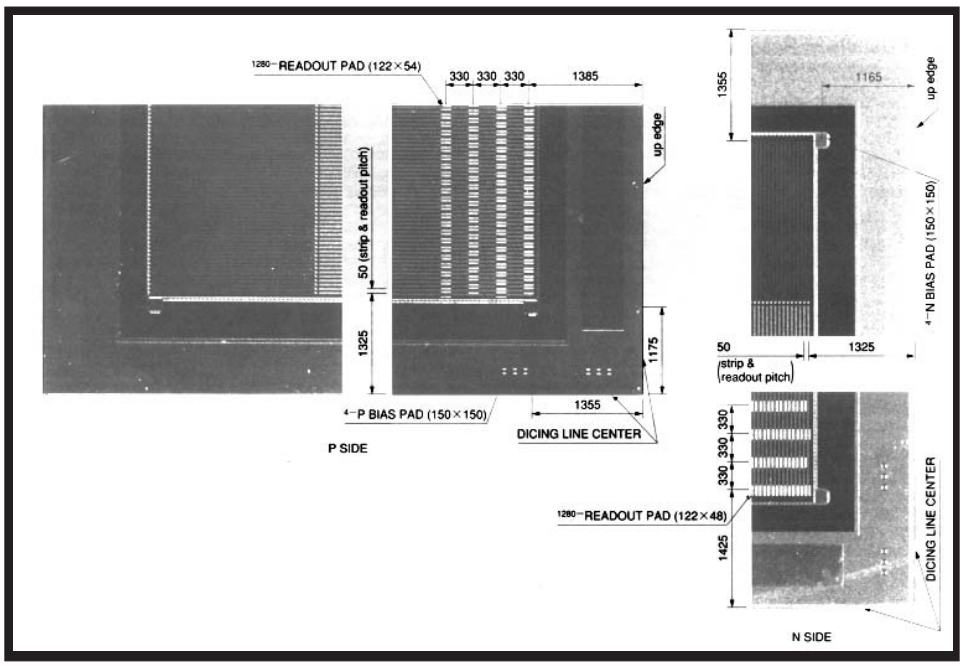


Figure 25: Details on the detector readout portion. Bond pad positions for strip and bias contacts relative for the chip edge are given. A spare bond pad for each strip contact is provided. The bias bond pads are large enough to hold at least 2 wire bonds contacting the bias voltage on the hybrid. The bias resistors are on the detector edge opposite to the detector strip contacts, so the bias voltage has to be routed around the detector once. To reduce problems with voltage drop on the bias lines, there is a bias voltage contact on each side of the strip contact bond pad row.

the readout pitch, no interstrips are present, although the case and the FE-electronics are suitable for sensors of equal size with interstrips, too. The strips are AC-coupled, each strip being biased with a polysilicon bias resistor. Galvanic decoupling for the N-side strips is provided by a p stop implantation. Table 5 gives an overview over the most important properties of the sensors. In figure 25 an overview about the detector outline, bond pad setup and bond pad location is given. More information on the detector type can be found in the Hamamatsu catalog on *Si photodiodes and charge sensitive amplifiers for scintillation counting and high energy physics*, catalog number KOTH00020E05, published in June 1997 by Hamamatsu.

6.2 Biasing the detector

Double sided silicon strip detectors are very sensitive to pickup on the bias voltage. A small voltage signal on the bias voltage with reference to the preamplifier ground injects charge via the strip coupling capacitance in all preamplifier channels in parallel. This is the so-called common mode noise, a fluctuation of all pedestal voltages by about the same value to positive or negative values, depending on the sign of the voltage step on the signal input. A very careful decoupling of the bias voltage is therefore necessary to maintain good system operation, as the BAT readout concept is very sensitive on common mode noise. The potential level of the detector strips has to remain stable relative to the reference potential of the VA input. The BELLE hybrids were originally thought to be operated with single sided silicon strip detectors (SSSD). When operating SSSDs, the detector back side, the side not divided into strips, can be directly connected to a large, low impedance bias contact. To avoid level shifting of analog or digital signals, this is usually the same ground potential the front end hybrids are connected to. The bias voltage has to be decoupled against this ground and drops over the coupling capacitance and over the detector diode. Although for this design pinhole channels can cause a group of VA channels to be dead[23], it has several advantages, especially the easy setup. Decoupling is done with a single low-pass filter on the hybrid close to the bond pad for the connection to the detector.
In the BAT, double sided strip detectors (DSSD) are used. As now the strips on both sides are high impedance nodes, the system is even more sensitive to common mode noise. The potential level of the strips has to remain stable with reference to the VA reference voltages on both sides. Figure 26 shows the detector bias scheme in the case the decoupling circuit described above and provided on the VA hybrids is used. This bias circuit was used on the ATC prototype design for the BAT. The main difference to the operation of SSSDs is that the VA ICs reference ground in the BAT setup is the bias voltage level itself, so there is no (or almost no) voltage drop over the coupling capacitors[24]. However, in this circuit the connection between ATC ground and bias ground is made on the ATC compartment directly behind the first RC low pass filter used to smooth the HV bias voltage. ATC and bias ground are not connected to each other on the hybrid. Thus, the wires connecting hybrid ground and hybrid bias with the ATC act as a large ground loop. The two RC low passes used to smooth the bias voltage when operating SSSDs are

[23] This is due to large amounts of leakage current flowing into them.
[24] This reduces the influence of pinhole channels.

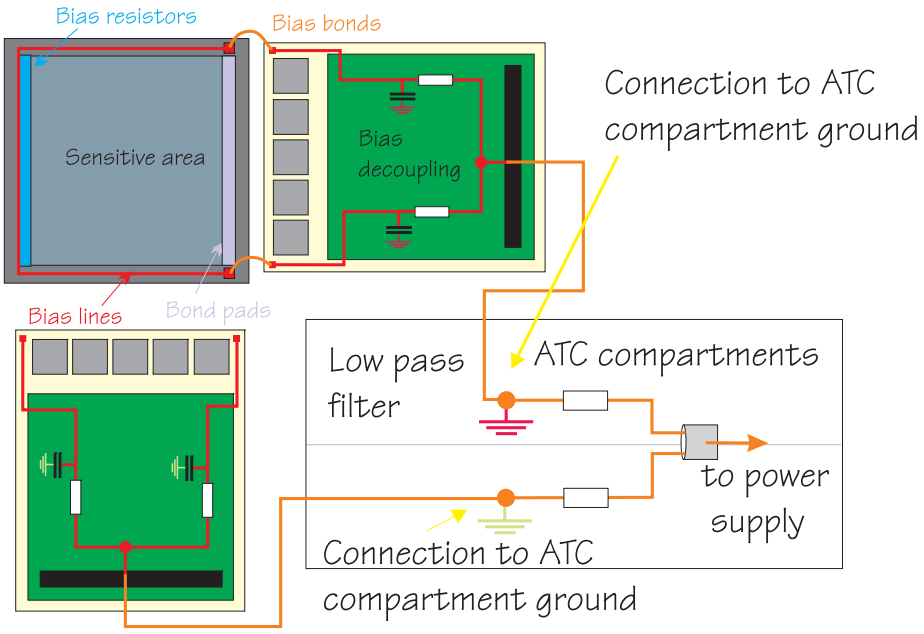


Figure 26: ATC prototype detector biasing scheme. The design is sensitive on common mode fluctuations, as the connection between ATC ground and HV bias is made too far away from the detector itself. The low pass filters are useless in this case, especially for low frequency common mode fluctuations.

useless in this case. Therefore, an improved circuit was used for the current version of the ATC. This circuit is displayed in fig. 27.

In this case, the connection between HV bias and ATC ground is made as close as possible to the detector itself. The HV bias is decoupled with reference to the potential level of the other detector side, not with reference to ATC ground. A second low-pass filter for HV decoupling directly at the detector was added. To realize this circuit, some modifications had to be made to the hybrids. The capacitors for the HV bias filtering low passes displayed in fig. 26 were bridged to connect ATC ground and HV bias. The two 100 kΩ bias resistors also belonging to these low passes were connected in parallel to form the series resistor of the second HV bias low pass filter. The large HV capacitor (1 μF) capacitor had to be glued to the aluminum frame (bricolage!) because there was only little space left in the case. A HV thin film capacitor had to be used, because a ceramic type with the matching capacitance was not available for this voltage range (up to 150 V), although it would have had a better high frequency behaviour. When implementing this second low pass filter, the resistors R1 and R2 have to be connected in parallel, and the capacitors C1 and C2 have to be bridged (numbers are given with reference to figure 24).

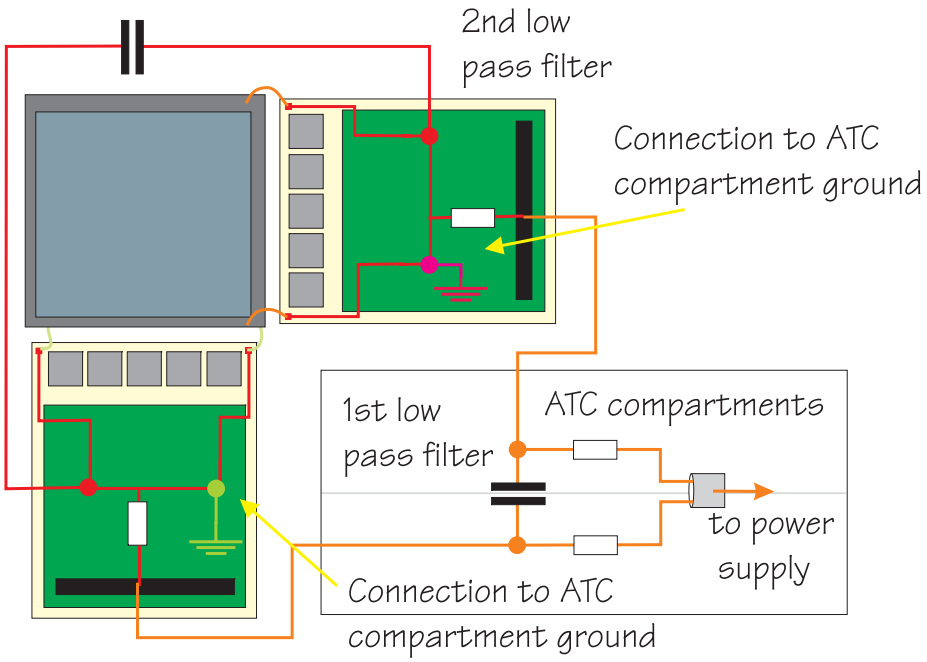


Figure 27: Final ATC detector biasing scheme. This design is less affected by common mode noise as the connection between HV bias and ATC ground is located closer to the detector.

Please note that making modifications to the hybrids like soldering on the PCBs, shortening or removing resistors or capacitors and so on should be done in any case before assembling a hybrid assembly. Due to the good thermal conductivity of as well hybrid carrier ceramics and assembly frame it is almost impossible to solder on a hybrid mounted into a correctly assembled hybrid assembly. All heat will flow directly to the assembly frame without melting the solder. If it cannot be avoided, one has to make sure that there are no fake solder joints and stuff.

7 The DTC

7.1 PCB features

The digital telescope card (DTC) is a double-layer PCB that holds all digital controls of one telescope module and provides an interface to the BB. Besides some passive components and drivers, the following parts are located on the DTC:

- The interface FPGA (IFPGA), mostly interface to the BB.
- The main FPGA (MFPGA), sequencer and data preprocessor.
- The module's FIFO (18 x 64 k, but only 16 x 64k are used).
- The module's RAM (16 x 64k)
- The coupler gate, 14 x 2 channels.

The details of the circuit schematics and layout can be taken from the documentation folder. A layout view of the card is presented in fig. 28, a photo of a DTC card in a module setup is given in fig. 29. A block diagram showing the functional building blocks is shown in fig. 30. The DTC BB connection is made via two differential drivers, the input/output signal lines are fed directly to the IFPGA. Located between drivers and BB connector there are sockets for resistors which can be used to terminate the BB. Please note, that in normal operation only the last module, the one that is connected to the end of the BB, has to be equipped with termination resistors[25]. A 5 item jumper between the two drivers is used to set the module's address on the BB. Only the left three jumpers are relevant for the module's address[26]. If the jumper is bridged, the corresponding bit is set to 0 with the leftmost jumper being the LSB. So, all three jumpers bridged corresponds to a module address 0, all three jumpers open corresponds to module address 7 and so on. Please note that the address 0 is already occupied by the PCI-BB interface card.

The three LEMO connectors on the right side of the card serve as input or

[25]Usually, this is the TLU. The TLU was given BB address 7 to remind the user that the TLU has to be the last module on the bus. When testing a single module it might be necessary to operate one module without a TLU (e.g. if no TLU is available). In this case, the socket can be equipped with appropriate terminators.

[26]In the following, alignment of components is described while holding the card with the coupler gate down.

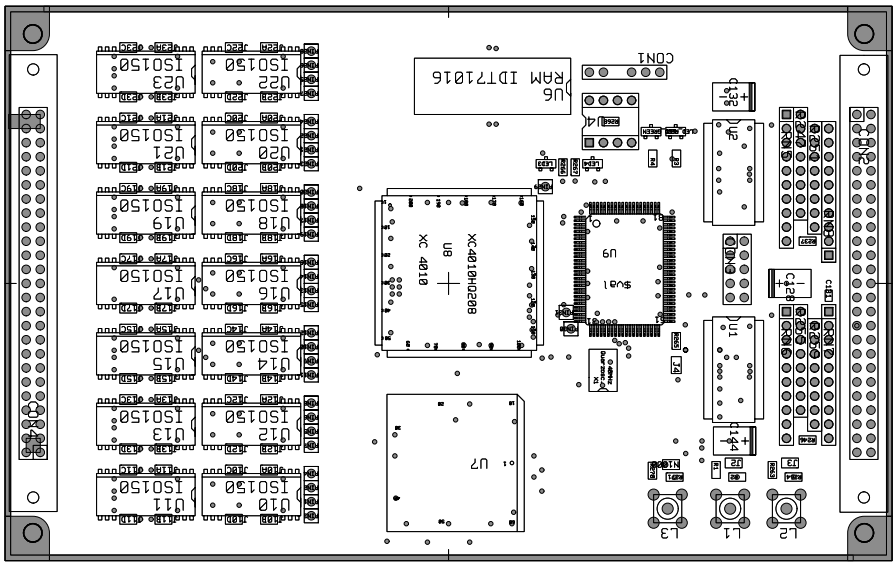


Figure 28: Layout of the DTC card showing the positions of the different components. The couplers' gate is on the left side, the connection to the BB on the right side. The RAM is on the upper side and The FIFO on the lower side of the PCB.

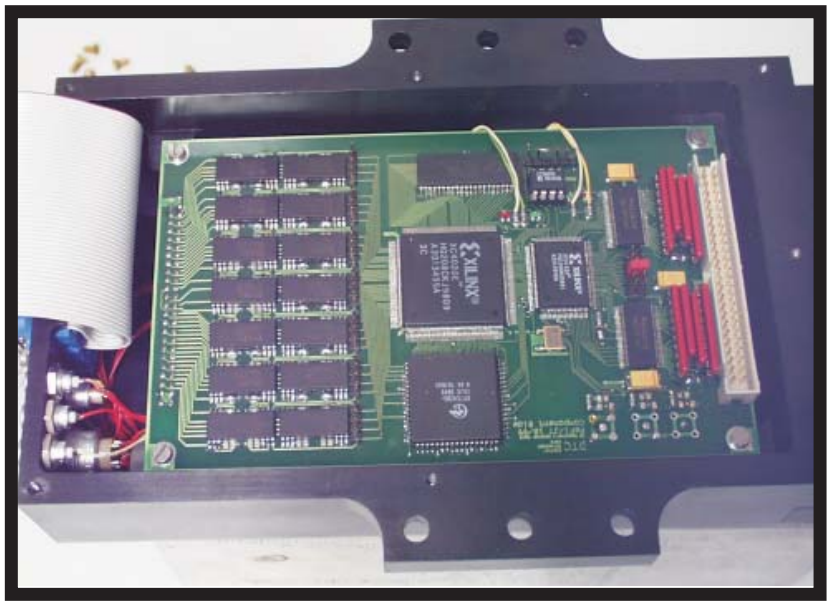


Figure 29: Photo of the DTC card in a module setup. The orientation is the same as in fig. 28. The flat ribbon cable used to connect the PCB with the case BB connectors has been removed.

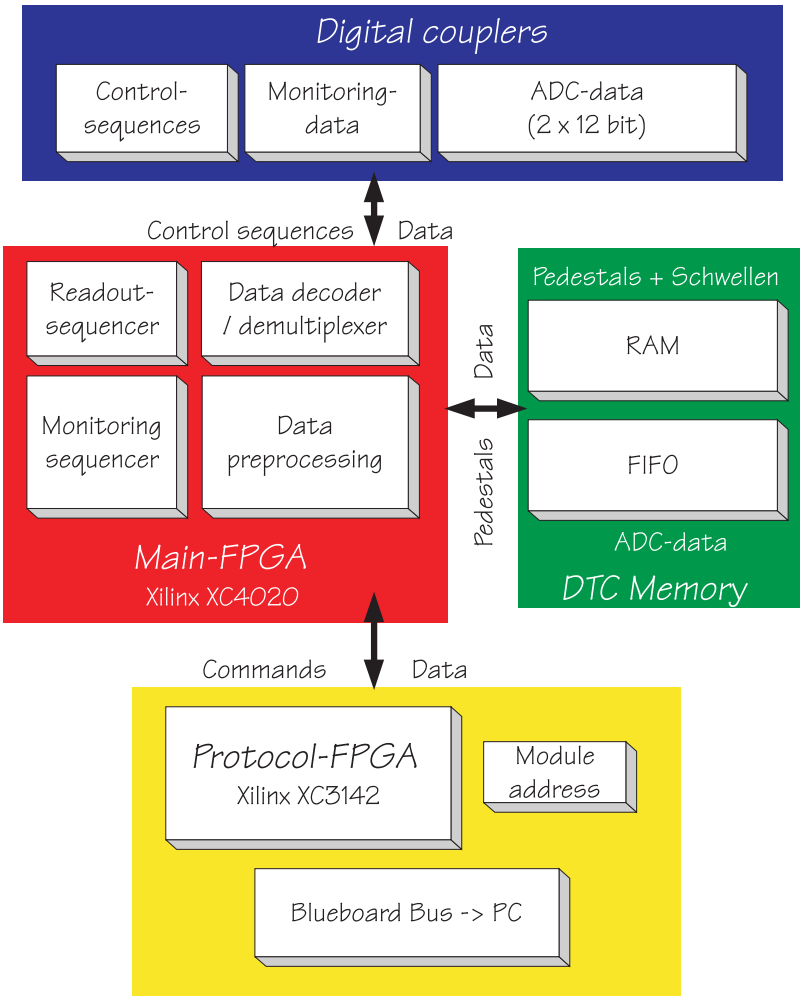


Figure 30: Functional building blocks of the DTC. The digital couplers make the connection to both ATC compartments. The DTC memory is mostly needed by the DPR, the FIFO for data storage and the RAM for storage of previously acquired and set pedestal and threshold values. The main FPGA contains all sequencer and data preprocessing functions, while the interface FPGA controls all accesses from and to the blueboard bus.

output for fast control signals (trigger etc.). LEMO 2 is an output, LEMO 1 and 3 are inputs. Close to the LEMO connectors there are pads for termination/pulldown and series resistors. At the moment all LEMOs are equipped with a 100 Ω series resistor and a 3 kΩ pulldown.
The system clock is generated by a 40 MHz crystal oscillator on the right of the IFPGA. On the left of the IFPGA, a socket for inserting a configuration PROM and a connector for a serial download cable are located. They are used for configuring the IFPGA after power up. If the jumper next to the

upper right corner of the IFPGA is bridged, the IFPGA will try to configure in master mode. In this case a configuration PROM has to be present, otherwise the IFPGA won't configure. If the jumper is open, the IFPGA can be configured in slave mode using a download cable. The configuration PROM has to be removed then. A green LED will indicate that the IFPGA has been successfully configured. IFPGA and MFPGA are connected via an 8 bit data bus, a 5 bit mode address bus, a three bit IRQ address bus and 5 control signals. Additionally, the 3 LEMO connectors are connected via the IFPGA directly to the MFPGA.
RAM and FIFO are only connected to the MFPGA. The FIFO is connected via 16 input lines, 16 output lines and few control signals. The FIFO once received its read and write clock from an output pin of the MFPGA, but due to timing problems it now gets its clocks directly from the oscillator[27]. The RAM is connected to the MFPGA via a 16 bit address bus, a 16 bit data bus and also some control signals. Both RAM and FIFO can be accessed via the MFPGA only using the corresponding control blocks.
The coupler gate is an array of 14 capacitive couplers, each providing 2 digital I/O channels. For each of the couplers, primary and secondary side are galvanically insulated. The primary sides of the couplers are all connected to the DTC, but the couplers serve as I/O connection to the two compartments of the analog telescope card. So there are 14 channels for which the ATC N-compartment is the secondary side. For the remaining 14 channels the ATC P-compartment is the secondary side. ATC N- and P-compartment are separated by the depletion voltage. From the 14 channels available for each compartment, 10 serve as control output from the DTC and the remaining four serve as data input from the ATC. Jumpers to the left and the right of each coupler allow the user to operate the two channels of each coupler either in send or in receive mode. Both of a coupler's channels are operating in receive mode, if the two jumpers on the left side of the coupler are closed and the two jumpers on the right side are opened. If the two jumpers on the right side of the coupler are closed and the two jumpers on the left side are opened, the coupler operates in send mode[28]. For each ATC compartment 5 couplers are operated in send and 2 in receive mode.
The four LED (two green and two red) are controlled by certain IFPGA and MFPGA status signals. The first green LED indicates correct configuration if the IFPGA. The first red LED indicated a pending IRQ generated by the module. The second green LED is controlled by the FIFO empty flag, the

[27] In the current version of the card, this connection had to be made using thin wires (bricolage), in an advanced version of the card this may change.

[28] It is important to change both sides simultaneously. Also jumper settings can neither be let open nor crossed.

second red is controlled by the FIFO full flag[29].

7.2 The interface FPGA

The interface FPGA (a XILINX XC3142-3 in a PQ 100 plastic case) is responsible for the correct handling of any kind of access to and from the BB. It decodes mode and data accesses, transmits mode or register data to the correct destination, answers strobe signals with acknowledges and so on. It is also aware of the module's bus address and allows accesses to this module's resources only if the address currently valid on the bus matches the module's bus address. Moreover, it contains some important system building blocks which will be described in the following:

- The central mode address register (CMA)
- The first mode decoder (MD1)
- The Interrupt status register (ISR)
- The main FPGA configuration register (COR)

7.2.1 The central mode address register (CMA)

The interface FPGA contains the module's central mode register, whose values can be accessed during a mode access. In the central mode register (8 bit) the current *mode address* is stored, that is, which register is to be accessed next. If a register's mode address is selected, the next write access will write into it and the next read access will read the register's contents. The MA once written to the CMA will persist until a new MA is written. So if it is necessary to read and write a register subsequently (e.g. if only one bit of the register contents has to be changed and the rest is to be kept), the mode address doesn't have to be rewritten then. The central mode address register is also responsible for the mode addresses of the main FPGA. The mode address is transferred to the main FPGA via a 5 bit mode address bus[30].

[29] When the DTC is mounted in a module's case, the PCB board LED won't be visible any longer, but they can be connected to case mounted LED. This has been done for the first five BAT modules, unfortunately the order of LED may have changed, so watch out.

[30] Although the central mode address register is a 8 bit register, only 32 mode addresses are accessible at the moment

7.2.2 The first mode decoder (MD1)

The first 4 mode addresses are reserved for modes in the interface FPGA. From the remaining 28 mode addresses for the main FPGA, 6 are unused. The 4 mode addresses for the Interface FPGA are decoded by the first mode decoder (MD1). All MAs can be set from software with the $*Mode = mask$ command, where *mask* is the MA value. Please find below a complete description of all mode addresses for the interface FPGA and their tasks.

1. MA 0: *Mode RDID* (read only)
   If this mode ist set, the implementation ID of the current configuration of the interface FPGA can be read. Used as a first communication check. as described in the section about the BB, the first 3 MSBs contain the class ID of a connected device, and the remaining 5 LSBs contain the revision ID. The current version of the IFPGA firmware has the revision ID 6, and the class ID of the DTC is 1.

2. MA 1: *Select ISR* (read / write)
   If this mode is set, the interface status register of this module can be accessed. It is read to find out which IRQ channels are enabled at the moment or if an IRQ on the BB was generated by this module. It is written to test the module's interrupt capability or to enable or disable module IRQ channels.

3. MA 2: *Select XCONFIG* (read / write)
   Needed to read and write the the main FPGA configuration register (COR).

4. MA 3: *Select REG2* (read / write)
   Using this register, the read / write DWORD accesses can be tested. It is not used in normal operation.

7.2.3 The interrupt status register (ISR)

The interface FPGA contains the interrupt status register (ISR), which controls the module's interrupt behavior.
A module can sent a BB IRQ in order to force the DAQ software to interrupt its execution and handle this IRQ request. One can think of many reasons why a module should send an IRQ, but at the moment, there are only four different IRQ conditions (IRQ channels) implemented, which are controlled by the ISR. The ISR is a 8 bit register. For each IRQ channel there is a

pair of 2 ISR bits, one of the 4 LSBs (the IRQ mask bit) and one of the four MSBs (the IRQ register bit), which are associated to this IRQ channel. The IRQ mask bit is used to enable or disable the channel, the IRQ register bit is automatically set if the corresponding IRQ channel becomes active. One of the four IRQ channels (channel 0) is reserved for the interface FPGA, the other three ones are used by the main FPGA. The main FPGA activates IRQ channels by setting a certain IRQ address on the 2 bit IRQ address bus between interface and main FPGA. The IRQ address is decoded with a 2 bit decoder[31], and the corresponding IRQ channel is activated, causing the appropriate IRQ register bit to become active. If the channel's IRQ mask bit is also set, the module will pull the BB IRQ line active, forcing the BB controller to process the IRQ. As there is only one IRQ line on the BB, the DAQ process which is handling the BB IRQ has to read the ISR contents of all modules connected to the BB and compare their IRQ register and IRQ mask bits to find out which module has sent the IRQ and what the reason for the IRQ has been.
An IRQ mask bit is set or deleted by using the $*Mode$ and $*Data$ commands:

$$\begin{aligned} *Mode &= SELISR; \\ *Data &= value; \end{aligned}$$

where $value$ is the new register value and $SELISR$ is the MA of the ISR. Please note: setting one or more of the LSBs (values 0x01 to 0x0f) will unmask IRQ channels, setting one of the MSBs (values 0x10 to 0xF0) will set one or more of the IRQ register bits. Setting both for the same IRQ channel (e.g. values like 0x11, 0x22 etc.) will immediately generate an IRQ request. If only one register bit is to be modified while the other ones are to be left untouched, it is recommended to do a ISR readback first. A command sequence to do this looks as follows:

$$\begin{aligned} *Mode &= SELISR; \\ value &= *Data; \\ value &= (value \mid mask) && \text{if the bit has to be set to 1 or} \\ value &= (value \;\&\; \sim mask) && \text{if the bit has to be set to 0} \\ *Data &= value; \end{aligned}$$

When processing the IRQ, first the module which caused the IRQ has to be determined. Next, the IRQC[32] has to be handled properly. For an error IRQ

[31] The default value of the IRQ address bus is 0. No IRQ condition is met then.
[32] *Interrupt request condition*, the condition which caused the module to ask for an IRQ.

for example, the error condition has to be removed, for an IRQ indicating lots of data the data has to be read back and so on. Then, the IRQ address has to be reset to zero if the IRQ was an IRQ generated by the main FPGA and, at last, the IRQ register bits can be cleared, forcing the IRQ to disappear. If the BB IRQ line is still set, it must now be due to another module's IRQ pending, and so the handling process has to continue its loop over all modules connected to the BB. The following list contains the information about IRQ channels and their IRQCs.

1. IRQ channel 0:
   Reserved for interface FPGA use. Not implemented yet.

2. IRQ channel 1:
   Event interrupt: The module's FIFO is not empty.

3. IRQ channel 2:
   Data interrupt: The FIFO data counter count value exceeds a certain threshold set by the user using the CMD II register in the main FPGA.

4. IRQ channel 3:
   Unused.

Which IRQ is generated under which circumstances is also specified in the section about data acquisition (DAQ).

7.2.4 The main FPGA configuration register (COR)

The main FPGA is configured in slave serial mode, the configuration sequence is generated by subsequently writing the corresponding bit patterns into the XCONFIG register of the interface FPGA, which is accessed using the Select XCONFIG mode address. The output pins of the interface FPGA connected to these register bits are connected to the main FPGAs configuration pins. Only 4 control signals have to be generated. Two signals, *reset* and *program* initialize the configuration sequence, *cclk* and *din* provide the configuration clock and data. If Bit 0 of COR is found to be set by the FPGA after writing the entire pattern, the configuration sequence was successfully completed. The configuration data is generated by the FPGA implementation software in a bitstream file format. A software tool embedded in the DAQ software allows the user to feed the configuration data directly to the main FPGA. This can happen at any time, after power-up or whenever a new configuration is needed. Please note: The interface FPGA should automatically configure

after power up, if a PROM with the correct configuration is attached and the jumper setting is right (see DTC hardware description for details). You won't be able to configure the main FPGA if the Interface FPGA is not configured.

7.3 The main FPGA

The main FPGA is the module's heart and brain. It contains not only all the logic needed to generate the monitoring and readout sequences, but also the entire data preprocessor. The data preprocessor performs pedestal subtraction, hit detection, readout sparsification and data compression. It also contains the IRQ generation logic, which allows flexible data buffering. Using data buffering, the readout speed is only limited by the sequencer run time. Thus, event rates in the kHz range can be processed. The main FPGA (an XILINX XC 4020E type in a PQ208 ceramic case for better heat conductivity) contains the following functional building blocks.

- The second mode decoder (MD2) for the decoding of the modes used in Main Xilinx
- The first (CMD I) and second (CMD II) command registers
- The RAM access block (RAC)
- The status bus readback block (STA)
- The software sequencer (SOSE)
- The main sequencer (MSQ)
- The data preprocessor (DPR)
- The shutdown state machine (SSM) and the interrupt control (IRQC)
- The FIFO control (FICO)
- The IO blocks (IOB)

The functionality of each of these items will be explained and discussed below.

7.3.1 Second mode decoder (MD2)

The second mode decoder MD2 decodes the remaining mode addresses (MA) 4-32 of the 5-bit mode address space. The mode addresses 0-3 are needed in the interface FPGA, so they are not available in the main FPGA.
Of these 28 mode addresses, only 22 are needed for BAT operation. All MAs can be set from software with the $*Mode = mask$ command, where *mask* is the MA value. Some of the MA are not used to read or write registers, but to provide a reset strobe to certain counters or registers. Those Modes have to be applied for a short time, but must be taken away afterwards. This can be done by resetting the MA to a "harmless" value (e.g. RDID or so). Please find below a complete description of all mode addresses and their tasks.

1. MA 4: *Reset triggernumber* (strobe)
   If this mode ist set, the trigger counter in the SSM is set to value zero. Important at the beginning of a new run.

2. MA 5: *Start* (strobe)
   If this mode is set and the system is in soft trigger mode (see CMD registers), a readout sequence is started by software. Needed for Pedestal acquisition.

3. MA 6: *Load* (write)
   Mode needed to set the hold delay value for the MSQ.

4. MA 7: *RWL* (read)
   Read LOW byte of the data word counter, which is part of DPR.

5. MA 8: *RDID* (read)
   Read ID. Mode needed to read back the ID of the current main FPGA implementation ID.

6. MA 9: *RWREG* (read / write)
   Mode needed to read or write the CMD I register.

7. MA 10: *RDSTAT* (read)
   Mode needed to read back the status bus.

8. MA 11: *RAMRDHI* (read)
   Mode needed to read back HIGH byte of current RAM addresses data contents. Only needed for test readback of RAM.

9. MA 12: *RAMRDLO* (read)
   Mode needed to read back LOW byte of current RAM addresses data contents. Only needed for test readback of RAM.

10. MA 13: *RAMWRHI* (write)
    Mode needed to write HIGH byte to current RAM address.

11. MA 14: *RAMWRLO* (write)
    Mode needed to write LOW byte to current RAM address.

12. MA 15: *RESRAMADDR* (strobe)
    If this mode is set, the RAM address counter, which is part of MSQ, is reset to value zero.

13. MA 16: *RWFIFO* (write)
    Mode needed to write a 16 bit word prepared in the FIFOHI / FIFOLO registers into the FIFO. This MA is also needed for the DWORD access, which transfers 4×8 bit from the FIFO to the PC.

14. MA 17: *RWFIFODATLO* (read / write)
    Mode needed to write the HI byte of a 16 bit word into FIFOHI register in FICO.

15. MA 18: *RWFIFODATHI* (read / write)
    Mode needed to write the LO byte of a 16 bit word into FIFOLO register in FICO.

16. MA 19: *BITBYTE 1* (write)
    Mode needed to set the lowest 8 bit of the SSQ.

17. MA 20: *BITBYTE 2* (write)
    Mode needed to set the bits 8-15 of the SSQ.

18. MA 21: *RDBATDATA* (read)
    Mode needed to read back data acquired by the software sequencer.

19. MA 22: *RWCONTROL* (read / write)
    Mode needed to read or write the CMD II register.

20. MA 23: *BITBYTE 3* (write)
    Mode needed to set the highest 4 bit of the SSQ.

21. MA 24: *RWH* (read)
    Read HI byte of the data word counter, which is part of DPR.

22. MA 25: *RESIRQ* (strobe)
    If this mode is set, the IRQ status is reset, so a new IRQ can be processed.

7.3.2 First command register (CMD I)

The CMD I register can be accessed with MA 9. CMD I is a 8 bit register. 7 of the 8 register bits are occupied. CMD I contains main sequencer control flags, trigger control and flags needed for test purposes. All register bits of CMD I can be set from software with the following set of commands:

$$\begin{aligned} *Mode &= RWREG; \\ *Data &= value; \end{aligned}$$

where *value* is the new register value and $RWREG$ is the MA of the CMD I. If only one register bit is to be modified while the other ones are to be left untouched, it is recommended to do a CMD I readback first. A command sequence to do this looks as follows:

$$\begin{aligned} *Mode &= RWREG; \\ value &= *Data; \\ value &= (value \mid mask) \quad \text{if the bit has to be set to 1 or} \\ value &= (value \;\&\; \sim mask) \quad \text{if the bit has to be set to 0} \\ *Data &= value; \end{aligned}$$

where *value* is an auxiliary variable and *mask* is the bit mask for the bit which is to be modified. If it is necessary to modify a group of bits which e.g. belong to the control of the same functional building block at the same time, *mask* is the bit mask not only for one bit but for all bits of the control group. If, for example, the bits 3,4 and 5 of the CMD I register belong to the same control group, the mask for the control group would be $mask = 0x38$, and the values written to the control group can be all combination of these bits. Please find below a complete description of all register bits and control groups of CMD I and their tasks.

1. *CMD I bits 0,1: SEQUENCER-MASK*. This is the control group which activates, stops and resets the MSQ with the following values:

   - SEQ-RESET (0x00): Resets all MSQ, FICO and DPR building blocks. Additionally, a RESET is given also to the module's FIFO itself.

- SEQ-RUN (0x01): MSQ is activated and waiting for trigger
- SEQ-STOP (0x02, 0x03): MSQ is inactive.

2. *CMD I bits 2,3: MAIN-TRIGGER-MASK.* This control group allows to select the trigger mode for MSQ data acquisition.Please use one of the values specified below:

   - MAIN-EI-TRIG (0x04): Enables external trigger signal from LEMO.
   - MAIN-IE-TRIG (0x08): Enables software trigger.
   - MAIN-NONE (0x00): No trigger is enabled.

   Care has to be taken not to activate both trigger sources accidentally at the same time. External trigger signals are used in normal DAQ. The main trigger for the telescope should always be distributed via the TLU if there is more than one telescope module to read out. This forces synchronous trigger behaviour. The internal trigger signal is generated by a mode command (MA 5) and used for pedestal acquisition.

3. *CMD I bit 4: LOAD FLAGS.* This flag is needed when programming the almost full / almost empty flags of the FIFO. If this flag is active and any data is written into the FIFO, it is interpreted as programming data for PAFF / PAEF values. See FIFO Datasheet or the description of the FIFO in this manual for more details.

4. *CMD I bit 6: MAIN-FS-SEQUENCER.* This switch decides, whether the slow signal sequences generated by the software sequencer or the fast signal sequences generated by MSQ are transmitted to the ATC. Slow sequencer signals are only needed to set DAC values and read monitoring ADC channels.

5. *CMD I bit 7: TESTMODE.* This flag is needed to externally write the FIFO for system test or flag programming purposes.

CMD I bit 5 is unused.

7.3.3 Second command register (CMD II)

The CMD II register can be accessed with MA 22. CMD II is a 8 bit register. 7 of the 8 register bits are occupied. CMD II contains preprocessor and interrupt generation control flags. Access to the CMD II register is possible using the same command sequences as for CMD I, using the corresponding

MA for CMD II instead of the one for CMD I. Please find below a complete description of all register bits and control groups of CMD II and their tasks.

1. *CMD II bit 0: THRESHOLDS-OFF.* This flag decides whether the pedestal subtracted analog data word is compared with the individual threshold value stored in system RAM or not. If the flag is not set, the digital discriminator produces an output signal, the so-called "hit flag", which signals the following readout whether the actual analog value exceeds its threshold value or not. If the flag is turned on, the hit flag for all data words is always set, independent from any threshold values. Additionally, the value of the hit flag is stored together with the data word within the FIFO (see "Output data format"). This process is called "discrimination" or "hit detection". Please note that an operation of the telescope without pedestal subtraction but with hit detection does not make sense, as the hit flag data read out contains no useful information.

2. *CMD II bit 1: SPARSE-OFF.* This flag decides whether the hit flag information obtained from the hit detection stage is used to store clusters in FIFO or not. If the flag is not set, the analog data words of only a channel which produced a hit signal and his four neighboring channels (two on each side) are written into the FIFO. This is called "sparsification" or "zero suppression". If the flag is set, all the analog data words of an event are stored together with their hit flag information in the FIFO. Please note that this switch has no effect if the hit detection is not active, and that hit detection only works correctly if "pedestal subtraction" is active.

3. *CMD II bit 2: PEDESTALS-OFF.* This flag decides whether the individual pedestal values stored in system RAM are subtracted from the decoded analog data values obtained from the ATC or not. This is called "pedestal subtraction". As the default is the operation with pedestal subtraction, the pedestal subtraction is turned off if this flag is set.

4. *CMD II bit 3: IRQC.* This flag controls the way a module generates interrupts. If this flag is not set, the module generates an interrupt after every trigger. The further data taking is blocked until the data of the IRQ-generating event has been read out. This modules busy signal is active until the data of the IRQ-generating event has been read out. This is called "event IRQ". If the flag is set, the module generates an IRQ only if the amount of data stored in the FIFO exceeds a number of

data words which can be set with the "IRQ rate bits" (see below). In this IRQ mode, which is called "data IRQ", the data taking is always active, independent of the IRQ status; it is only blocked if the modules emergency flag indicates insufficient capabilities to store further evens (see "Data IRQ").

5. *CMD II bits 4, 5: IRQ-RATE.* This control group decides about IRQ threshold, the amount of data words which can be written into the modules FIFO until an IRQ is signaled. When the amount of FIFO data exceeds the value specified by the IRQ-RATE bits, the IRQ is generated after the processing of the actual event has been finished. The amount of data words stored in FIFO at the moment of IRQ generation is stored in a separate 16 bit register, which can be read out using MA 7, RWL, and MA 24, RWH. Usually, this register is read back before the transfer of the data from the FIFO to the controlling PC starts, so the DAQ knows the amount of data which is to be read back[33] So, the IRQ-RATE determines the frequency a module generates IRQs, which is important as the IRQ frequency on the BB must not become too high. There are four categories:

   - *Value 00: IRQ-RATE-0.* The IRQ is generated after at least 1024 words have been written into the FIFO. As the normal event consists of 16 words, 64 events fit into the FIFO at this IRQ-RATE before an IRQ is generated.
   - *Value 01: IRQ-RATE-1.* 4096 words are needed to generate an IRQ, which corresponds to approximately 256 events.
   - *Value 10: IRQ-RATE-2.* 16384 words are needed to generate an IRQ, which corresponds to approximately 1024 events.
   - *Value 11: IRQ-RATE-3.* 32768 words are needed to generate an IRQ, which corresponds to approximately 2048 events.

   The IRQ-RATE should be chosen corresponding to the trigger rate and the rate the system can process IRQs with. The IRQ frequency is not directly connected to the IRQ-RATE value, as the event size (the amount of data words of which an event consists) depends on the readout mode (e.g. on the SPARSE-OFF option or the COMPRESS option) and simply on the data quality (number of clusters per event, size of clusters). See the section about IRQC for more details.

[33] Additionally, this ensures that only complete events are transferred to the PC.

6. *CMD II bit 6: COMPRESS-OFF.* This flag decides whether the FIFO write block writes strip addresses for each cluster strip into the FIFO or if only the address of the first strip in the cluster is written. If the flag is set, for each strip in the cluster a strip address is written. If the flag is not set, the strip address is written only for the first strip in the cluster. This is the normal readout mode, as it reduces the amount of data which is to be read back in the most efficient way. The other mode is useful for debugging.

CMD II bit 7 is unused.

7.3.4 The RAM access block (RAC)

The RAC allows the access to the system's RAM. The RAM is used to store pedestal and threshold values, which will be used by the DPR while processing an event. The RAM cell currently activated is addressed by the RAM address counter (implemented in MSQ macro). The RAM is written byte by byte. For every RAM write or read access the RAM address counter is incremented, so one has to write either all high bytes first before starting to write all the low bytes[34]. The Hi/Lo bytes are accessed by using MA 13/14; selecting one of these and performing a write data access will write the data into the Hi/Lo byte of the cell currently addressed by the RAM address counter. The RAM address counter is automatically incremented then. There is no way to randomly access cells in the RAM. The counter can only be incremented by 1 or reset (using MA 15). For test purposes, the RAM contents can be read similarly by e.g. reading all Hi bytes first and all Lo bytes afterwards.
RAM access sequences look as follows.
Reading a HI byte:

$$\begin{aligned} *Mode &= RAMRDHI; \\ value &= *Data; \end{aligned}$$

Writing two values to the LO bytes of subsequent RAM addresses:

$$\begin{aligned} *Mode &= RAMWRLO; \\ *Data &= value1; \\ *Data &= value2; \end{aligned}$$

[34] Of course, one can write all Lo bytes first and then write all Hi bytes.

Cell #	Contents bits 15-12	Contents bits 11-0
4	4	0
5	4	0
6	Threshold channel 0 P	Pedestal channel 0 P
7	Threshold channel 0 N	Pedestal channel 0 N
8	Threshold channel 1 P	Pedestal channel 1 P
9	Threshold channel 1 N	Pedestal channel 1 N
10	Threshold channel 2 P	Pedestal channel 2 P
11	Threshold channel 2 N	Pedestal channel 2 N

Table 6: Example for RAM contents. The cells 4 to 11 are shown. This is the canonical order, in which values are written into the RAM. The values are made available for DPR automatically by the MSQ.

In normal operation, the user prepares a RAM content with appropriate threshold & pedestal settings e.g. calculated from some dummy events without hits. These are written to RAM before starting a run. When the system receives a trigger, the MSQ will operate the RAM address counter. It will make sure that the correct threshold & pedestal values will be applied to their corresponding channel. Please note: As the MSQ starts much earlier than the DPR, there are some "dummy offset" RAM cells in which no useful information is stored. So P&T data for channel 0 is not stored in RAM address 0 as one might assume. Pedestal and threshold values for channel 0 have to be stored in cell 6 for the P-compartment and in cell 7 for the N-compartment (the channel data for the same channel number on N- and P-side is stored in subsequent RAM cells). The pedestal values of the dummy offset cells should be set to 0 and their thresholds should be set to 4. As the 12 LSBs of a RAM cell belonging to a channel are the pedestal value and the 4 MSBs correspond to the thresholds, the RAM content looks like shown in table 6.

For extremely long cable length ($\approx$ 26 m)an error has been observed while reading back the RAM. This is due to a timing error in readback. Writing P&T data is not affected, so DPR will work correctly. The error will be fixed in a later version of the MFPGA configuration.

7.3.5 The status bus readback block (STA)

The status bus is a 8 bit register which contains the most important status signals. They are continuously updated and provide information about the

module's status. The status bus can either be read back using MA 10 or while MA 16 (RWFIFO) is active, the latter because the FIFO flags are also part of the status bus and one might want to look at them during FIFO readback without having to perform two additional Mode accesses[35]. A description of the status bits and their meaning follows.

- *Status Bit 0: Busy*
  The busy signal indicates that the MSQ is currently active and processing a trigger.
- *Status Bit 1: FIFO full*
  All 64 k FIFO cells are occupied with data. this condition should not occur in normal operation.
- *Status Bit 2: FIFO empty*
  The FIFO is empty, all data has been read out or no data was written yet.
- *Status Bit 3: FIFO almost empty*
  The FIFO programmable almost empty flag can be programmed to any value between 0 and 32 k. The flag indicates if the FIFO fill level is below this mark. Needed for the FIFO readout architecture, has to be set to 1.
- *Status Bit 4: FIFO almost full*
  The FIFO programmable almost empty flag can be programmed to any value between 0 and 32 k. If there is less space in the FIFO left, it will be indicated by this flag. Is used as "emergency" flag.
- *Status Bit 5: Data interrupt*
  One of the bits that activate an IRQ channel. Was used for debugging.
- *Status Bit 6: Old*
  Important flag for FIFO readout. Will be explained in context with the FIFO readout architecture.
- *Status Bit 7: Data IRQ flag*
  This bit generates the shutdown state machine data interrupt. Was used for debugging, is no longer important.

[35] This feature is not used in the current version of the telescope software.

7.3.6 The software sequencer (SOSE)

The software sequencer was first used for the software controlled system tests[36]. It is also used for programming the ATC DAC and reading back the ATC monitoring ADC. And it is used to set clock control modes and chip selects for the ATC components. The SOSE consists of 3 registers, each containing 8 bit, in which one register cell corresponds to one control or data output to the ATC. The outputs to N- and P- compartment of the ATC can be controlled independently (this is different for the MSQ). Some of the SOSE sequencer bits are controlled by the MSQ if in fast sequencer mode, others can only be controlled by the SOSE. For an overview about the programming sequences for the ATC DAC and the readout sequences for the monitoring ADC please refer to their datasheets[37]. An access to one register bit is equivalent to any CMD register access which only changes one bit of the register contents and leaves the rest untouched. But as the SOSE registers cannot be read back, the user has to remind the actual values and write the modified value without having read the actual register contents. This looks as follows:

$$
\begin{aligned}
*Mode &= BITBYTE \quad 1 \quad (2,3); \\
value &= (IntValue \mid mask); \qquad \text{if the bit has to be set to 1 or} \\
value &= (Intvalue \;\&\; \sim mask); \qquad \text{if the bit has to be set to 0} \\
*Data &= value; \\
IntValue &= value; \qquad \text{Remember new register contents here}
\end{aligned}
$$

with IntValue being the internally stored register contents. Please find below a list of all bits and their assignment to output pins.

- Register 1
  - *SOSE I bit 0:* CLK P
    Global ATC clock P side.
  - *SOSE I bit 1:* S 0 P
    Output multiplexer control bit 1 P side.
  - *SOSE I bit 2:* S 1 P
    Output multiplexer control bit 2 P side.
  - *SOSE I bit 3:* HLD P
    Hybrid hold signal P side.

[36] Meanwhile, this task has been transferred to the main sequencer MSQ.
[37] Both, ADC and DAC are programmed / read out serially.

  - *SOSE I bit 4:* STR P
    Calibration chopper strobe P side.
  - *SOSE I bit 5:* DTI P
    Global Data in P side.
  - *SOSE I bit 6:* DRT P
    Hybrid reset P side.
  - *SOSE I bit 7:* CLK N
    Global ATC clock N side.

- Register 2
  - *SOSE II bit 0:* S 0 N
    Output multiplexer control bit 1 N side.
  - *SOSE II bit 1:* S 1 N
    Output multiplexer control bit 2 N side.
  - *SOSE II bit 2:* HLD N
    Hybrid hold signal N side.
  - *SOSE II bit 3:* STR N
    Calibration chopper strobe N side.
  - *SOSE II bit 4:* DTI N
    Global Data in N side.
  - *SOSE II bit 5:* DRT N
    Hybrid reset N side.
  - *SOSE II bit 6:* TON P $*$
    Hybrid test mode P side.
  - *SOSE II bit 7:* CTR 0 P $*$
    Global clock control switch bit 1 P side.

- Register 3
  - *SOSE III bit 0:* CTR 1 P $*$
    Global clock control switch bit 2 P side.
  - *SOSE III bit 1:* TON N $*$
    Hybrid test mode N side.
  - *SOSE III bit 2:* CTR 0 N $*$
    Global clock control switch bit 1 N side.
  - *SOSE III bit 3:* CTR 1 N $*$
    Global clock control switch bit 1 N side.

- *SOSE III bit 4:* unused
- *SOSE III bit 5:* unused
- *SOSE III bit 6:* unused
- *SOSE III bit 7:* unused

The bits marked with an asterisk $*$ can be operated by the SoSe only. Even while operating in fast sequencer mode, the values written in the SoSe registers will be relevant for these bits. CTR 1 and CTR 2 serve two purposes at the same time. First, they are used to set the clock destination on the ATC (not all devices on the ATC which need a clock receive a clock all the time). And second, they serve as *chip selects* for the components which are selected to receive a clock. More on this in the section about the ATC.
Of course, when operating the module in SOSE mode, there has to be a way of reading back data from the ATC software controlled, e.g. when reading back DAC / ADC data. For this purpose, MA 21 can be used. How to use this feature will be explained in the ATC description.

7.3.7 The main sequencer (MSQ)

The main sequencer is implemented as a state machine in verilog HDL. It provides all the control signals for data acquisition, data preprocessing and the communication between the module and the TLU. It also takes care for the correct storage of telescope data in the module's FIFO. When the MSQ is active, control over the system's RAM will be transferred to it. The MSQ takes data[38] relevant for the data preprocessing out of the RAM and makes sure that it is in place right in time. The MSQ controls both N- and P-side compartment of the ATC simultaneously.
The control sequence generated by the MSQ is essentially the basic VA readout sequence fitted out with some additional signals controlling RAM access, data preprocessing, digitization, multiplexing and demultiplexing of data. For the VA readout sequence please refer to the documentation about front end electronics. The MSQ generates essentially the same readout sequence for both N- and P- compartment of the ATC, so all VA circuits have to be

[38] Pedestals and threshold values.

tuned to the same hold delay value. The hold delay value is the only parameter which can be influenced by the user. The hold delay value is set for both compartments simultaneously by using MA 6. The hold delay value can only be adjusted in steps of 25 ns, with a 25 ns jitter[39]. The hold delay value can be adjusted between 0 and 255, which corresponds to a hold delay time between 320 ns (a build-in offset) and 6.68 μs. An overview about the structure of the main sequencer and its position in the MFPGA is given in figure 31.

The MSQ starts when the start condition is met. The start condition is met, when the following situations occur:

- *Software trigger condition*
  The system operates in fast sequencer mode (MAIN-FS-SEQUENCER bit (CMD I bit 6) is set). The MSQ is activated (SEQUENCER-MASK (CMD I bits 0 and 1) is set to value $0x01$). The trigger source is set to software trigger (MAIN-TRIGGER-MASK, CMD I, value $0x08$) and a start strobe is given (by writing MA 5, a strobe address[40]).

- *Hardware trigger condition*
  The system operates in fast sequencer mode. The MSQ is activated and the trigger source is set to hardware trigger (MAIN-TRIGGER-MASK, CMD I, value $0x04$). The INHIBIT signal (connected to LEMO 2) has to be inactive. The ARM-Trigger signal has to be active.

If the MSQ is busy processing one trigger, it won't be able to process another one. There is no such thing like pipelining of hits or triggers[41]. The module uses an internally generated BUSY signal, which is active while the MSQ is running, to block further triggers. New triggers will only be passed to the MSQs input stage if the BUSY signal is inactive. To enforce synchronous triggers for all modules the BUSY signal is used to generate a MODULE-BUSY, which is transferred via LEMO 1 to the TLU where, in coincidence with the other module's MODULE-BUSY signals, the decision is made if a new trigger is to be generated or not[42]. The ARM-Trigger signal is only important while using the hardware trigger. The condition, on which the ARM-Trigger signal is active, depends on the IRQ mode the module is operated in. In DIRQ mode, a module can take a new trigger if the old trigger

[39] This can, under some circumstances, cause signal loss. This loss is, however, negligibly small compared to the total signal amplitude. The shaping time is about 2 orders of magnitude larger than the 25 ns jitter, and at the peak the signal lapse is flat.

[40] To a strobe type mode address, a register is not connected.

[41] This is due to the VA 2 readout architecture.

[42] More on that can be found in the section about the TLU.

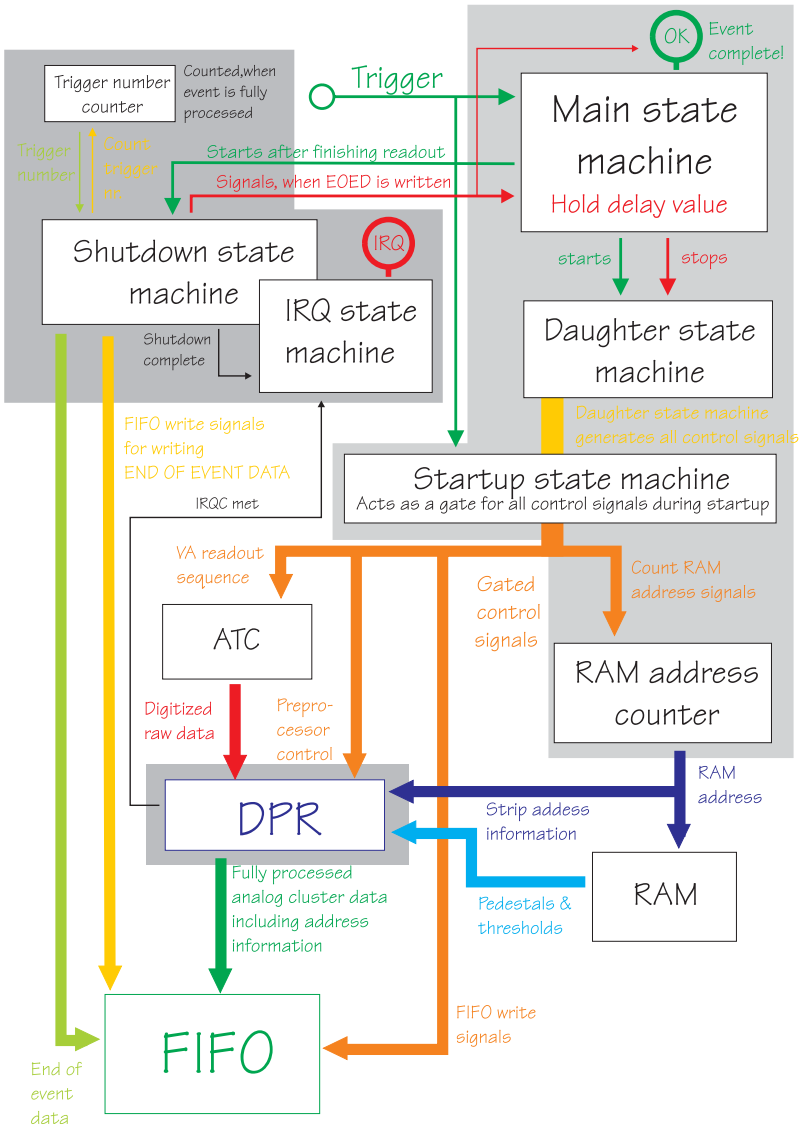


Figure 31: Overview about the MSQ position in the MFPGA structure. The blocks underlaid with gray are contained in the MFPGA, the other ones are external DTC components. The large gray block to the right is what is called MSQ in the text. It is composed of 4 smaller HDL macros. The interaction between the blocks is also shown. For a functional description please refer to the sections describing the corresponding blocks, for a more detailed documentation please refer to the source code files in the documentation folder.

has been fully processed, and a trigger is fully processed if the MSQ has completed a full cycle. As the MSQ's BUSY signal will block any further triggers if the MSQ is running anyway, in DIRQ mode the ARM-Trigger signal is always active. In EIRQ mode things are a bit different. The problem is that the event is fully processed, if the data taken for the event has been read back completely. No further triggers are to be accepted until this condition is met. Thus, the ARM-Trigger signal is set active by the RESIRQ-access (MA 25) which is performed by the DAQ software after reading back a data block[43].
The MSQ is set up to acquire, process and store any data coming from the 640 VA 2 hybrid channels[44]. The system clock is generated by the crystal oscillator on the DTC. Its frequency is 40 MHz, corresponding to a period of 25 ns. The output data has to be 4×4 multiplexed and all multiplexer signals have to be generated by the MSQ. To achieve greater timing stability all MSQ signals are synchronized at the MFPGA output pins[45], and this is done with the system clock. From these boundary conditions it is easy to see that the maximum frequency of the MSQ output signal can only be 20 MHz[46]. Due to 4 to 1 multiplexing, the conversion clock frequency has to be lower by a factor of 4, that is 5 MHz. Thus, the MSQ needs 200 ns to process one channel, so the total MSQ runtime is

$$200\text{ns} \quad \times \quad 640 \quad = \quad 128 \quad \mu\text{s}$$

In fact, there are 1280 channels as the telescope modules hold double sided strip detectors (so two hybrids have to be read out), but as both sides are processed in parallel, the MSQ runtime is not influenced. After finishing processing of all the channels, the MSQ does not simply stop. There is additional data besides the cluster / address data taken from the hybrids which has to be made available for the DAQ. This data is written by the shutdown state machine (SSM). The SSM is started by the MSQ with a *shutdown start* signal after the MSQ has finished processing data. The MSQ waits until a *shutdown complete* signal from the SSM indicates that the shutdown of the event has been completed. Then, finally, the MSQ finishes, de-activates the BUSY signal and prepares for the next trigger before it starts waiting for the next trigger to occur. As this takes some time and furthermore at the beginning of an event the MSQ waits for the hold delay to pass, the total MSQ runtime sums up to 135 μs (the exact value depends on the hold delay

[43] A more detailed description of the IRQ operation modes is given in the section about interrupt control (IRQC).
[44] 640 channels per detector side, of course!
[45] See section about the IOBs.
[46] This demand arises from the Shannon-Whittaker sampling theorem.

value), corresponding to a maximum possible rate of processed events of 7.4 kHz.

7.3.8 The data preprocessor (DPR)

The data preprocessor is the building block of the MSQ which holds all data preprocessing functions. There are 5 major tasks:

- Data demultiplexing (DD)
- Pedestal subtraction (PS)
- Hit detection (HD)
- Zero suppression (ZS)
- Data compression (DC)

Besides, there is another important functional block implemented in the DPR. This is the output multiplexer (OMU), which takes care of the correct writing of preprocessed data into the FIFO. And, at last, the *common mode counters*, whose data allow an off-line correction of common mode noise, are closely related to the DPR circuit.

Data demultiplexing. Actually, data demultiplexing is not really a data preprocessing task. The data coming in from the ATC compartments is 4×4 multiplexed and has to be demultiplexed anyway. As the data acquisition of ACT data runs serially, one has to deal with a 640 12-bit-word sequence of ADC data for each side. The data is processed step-by-step. As the data acquired by the fast ADCs on the ATC is only 12 bit wide, 4 bits of the 16 bit information obtained from the ACT compartments can be thrown away[47]. The remaining 12 bit are converted to parallel data words by the DD stage and, for simplicity, expanded to a 16 bit word by simply setting the upper 4 bit to 0. In the following, the processing of N- and P-compartment data of ATC data runs in parallel. The DD is now complete and the demultiplexed data available at the input of the PS stage.

[47] More about the ATC output data format in the ATC section.

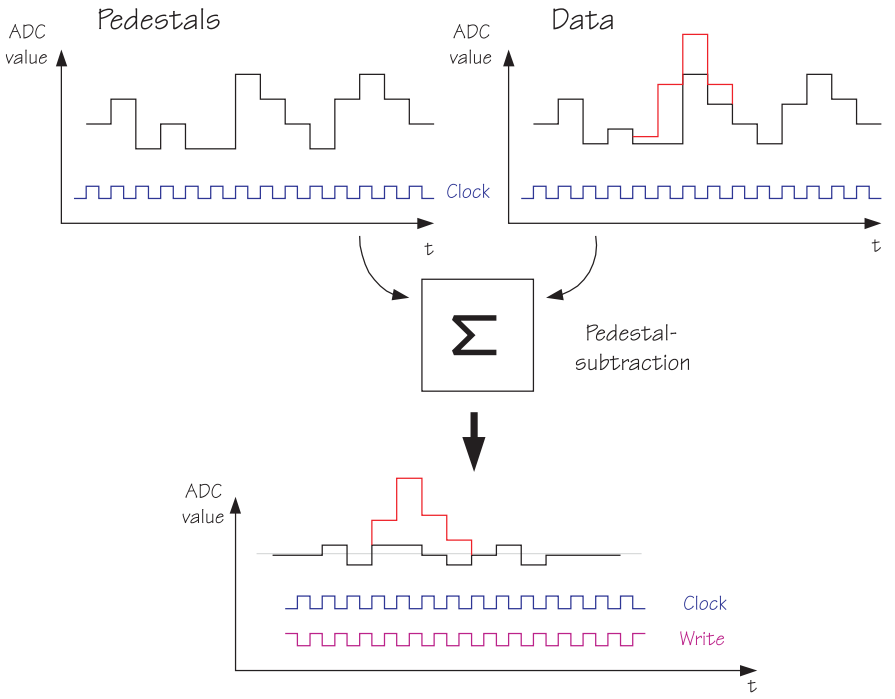


Figure 32: The principle of pedestal subtraction. For each channel, a pedestal value is acquired during run preparation. This value is stored internally. During event data processing each channels pedestal value is automatically subtracted from the data acquired from this channel during the event. Due to pedestal fluctuations, the result is not a "zero line", but distributed around zero, except for hit channels, which will stick out.

Pedestal subtraction. A schematic view of the pedestal subtraction operation is shown in fig. 32. If the demultiplexed data for channel n, v_n, arrives after conversion clock cycle (CCC) c_n at the input of PS, the MSQ controlling the RAM access makes sure that the pedestal value for channel n, p_n, is also available there at the same time. With the next CCC, c_{n+1}, the PS stage performs the operation

$$cd_n \;=\; v_n \;-\; p_n$$

Care has to be taken here not to mess up the signs of the pedestal corrected data. As for a channel without a hit v_n can be smaller than p_n due to pedestal fluctuations, the difference can be negative. As the 16 bit adder / subtractor implemented in the FPGA treats negative numbers as two's complement, one has to take care that the information about the sign of the result is not lost. This information is contained in bit 15 of the result (which was formerly unused). This bit decides how to interpret the 12 bit value cd_n correctly as

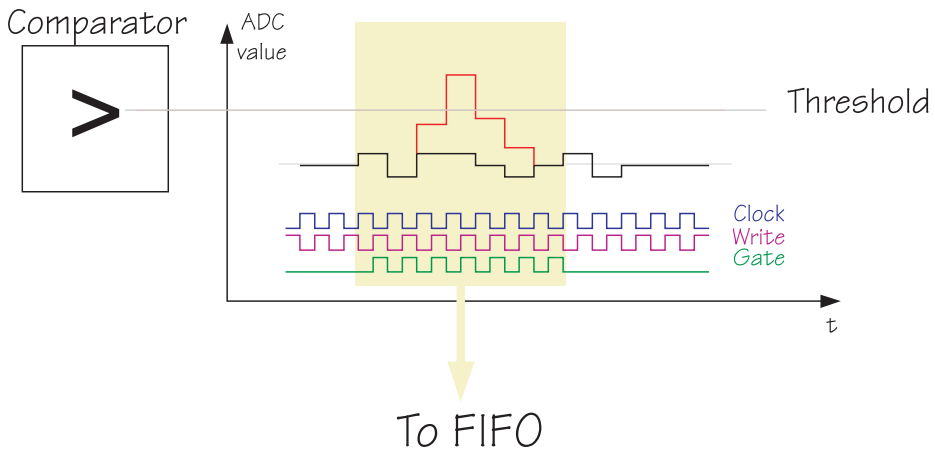


Figure 33: Principles of hit detection and zero suppression. The pedestal corrected values are compared with an individually set threshold value. If a channel exceeds this threshold, its *cd* value and the values of the neighboring channels are written into the systems FIFO.

well during hit detection as in online / offline monitoring. Then, with c_{n+2}, cd_n is transferred into a pipeline shift register, in which this data word will stay for 5 CCCs. At the same time, the HD stage will take care about the data word.

Hit detection. The principle of Hit detection (together with zero suppression) is shown in figure 33. In the HD stage it will be decided whether the pedestal corrected data word cd_n exceeds a threshold value t_n, which is encoded in a 4 bit word. t_n makes up the uppermost 4 bits of the RAM cell in which the pedestal word p_n is stored. t_n is made available at c_{n+2} at the input of HD by the MSQ, which also took care about p_n arriving at c_{n+1} at the input of the PS stage. If c_n exceeds its threshold, a "hit bit" will indicate that the threshold condition was met and mark the channel for the following DPR steps, online and offline monitoring. The hit bit is part of the analog output data word. The hit bit replaces bit 13, which so far was 0. A channel whose hit bit is set to 1 will in the following be called a *hit channel*.

The value t_n is a coarse threshold category rather than an exact threshold value. Although it is a 4 bit value, it can only take 5 values. The corresponding bit pattern, which is to be written into RAM to have the threshold applied, is noted in brackets:

- Value 4 (0x8): The channel n will never produce a hit, independent from

the value of cd_n[48]. Please note: This *only* means that this channel will neither be marked as hit channel nor prompt the DPR to write a new cluster to FIFO. This does *not* mean that cd_n will never be written into FIFO. In fact, this can happen, if channel n is part of a cluster caused by a neighboring channel e.g. if channel n+1 or channel n-2 produced hits. So, care has to be taken not to misinterpret a masked channel, which is masked because it is noisy, as a hit[49].

- Value 3 (0x4): The channel will produce a hit, if cd_n is larger than 512 ADC counts. If compared to the usual signal amplitudes and noise values, this can be called a "high threshold".
- Value 2 (0x2): The channel will produce a hit, if cd_n is larger than 256 ADC counts. This is a "normal threshold".
- Value 1 (0x1): The channel will produce a hit, if cd_n is larger than 128 ADC counts. This is a "low threshold".
- Value 0 (0x0): The channel will produce a hit independent from the cd_n value[50]. This feature is used for debugging and to produce dummy data of a realistic size in bench-top tests of the DAQ and OMO software.

If, for example, running with HD but without ZS it will be possible to determine hit channels by looking at the hit bits[51].

Zero suppression & data compression. The principle of zero suppression is shown in figure 33. In conventional telescope systems all the analog data acquired from the VA circuits after digitization has to be transmitted to the DAQ software. Hit detection and pedestal correction have to be done offline. The BAT system can easily perform a ZS if the HD process is completed. The hit bit obtained from the HD stage will be used to generate a *write gate* signal. The write gate signals the OMU which data is to be written into the FIFO and which is not. Zero suppression here does not mean that only those *cd*s are stored which exceed their threshold. Zero suppression means that for each hit channel indicated by a hit bit a larger group of neighboring channels, a *cluster*, will be written.

[48] Such a channel will, in the following, be referred to as "masked channel".

[49] The most effective way to do this is by comparing the cd_n read back with the thresholds.

[50] Such a channel will, in the following, be referred to as "unmasked channel".

[51] The same information could be gathered by comparing cd_n with t_n without looking at the hit bit.

Principle of operation. The telescope readout is analogue, because the analog information is to be used for a very precise reconstruction of the position where the ionizing particle crossed the telescope module's detector. To perform this task, not only the *cd*s which exceed the threshold are needed but also the *cd*s of their neighboring channels. Thus, the write gate causes the *cd* values of a cluster to be stored. In this context a cluster in its simplest form consists of the *cd*s of one "hit channel", a channel, whose hit bit is set, and the *cd* of a number of neighboring channels on either side. This number is called *cluster range*, and in the present DPR it is set to 2.
In addition to the analog values themselves, a position information has to be stored. This position information is needed to determine the position of those channels on the detector which detected the cluster.[52]. As one channel is processed within one CCC, e.g. the number of CCCs could be taken as an address information. More on that later.
Depending on the selected acquisition mode, there are two different formats in which this information can be provided:

- The address information is stored for each channel in the cluster. This option requires more storage space and contains redundant information, but provides many possibilities of integrity checks. This is the *normal* mode.

- The address is only stored once for the first channel in the cluster. As the other analog values should belong to subsequent channel numbers, the assignment of the following analog values to channel numbers should be possible. This option saves buffer space in the FIFO and is to be preferred especially when operating with a high intensity beam or high trigger rates. This is the *compressed* mode. So the DC option reduces the data size in the FIFO by nearly a factor of two by simply omitting redundant address information.

In figure 34 both compressed and non-compressed events are shown. As explained above, the DC option is not a full preprocessing step but a way to store address information saving storage space. If the DC option is selected, the addresses will be written in the compressed mode, if not, in the normal mode.
It was said above that a cluster in its simplest form consists of five neighboring channel's *cd*s accompanied by some address information, the hit channel

[52]This in not needed in common telescope architectures. There, the position on the detector can be obtained by simply counting the discrete ADC values read back from the modules, each value belonging to a different channel.

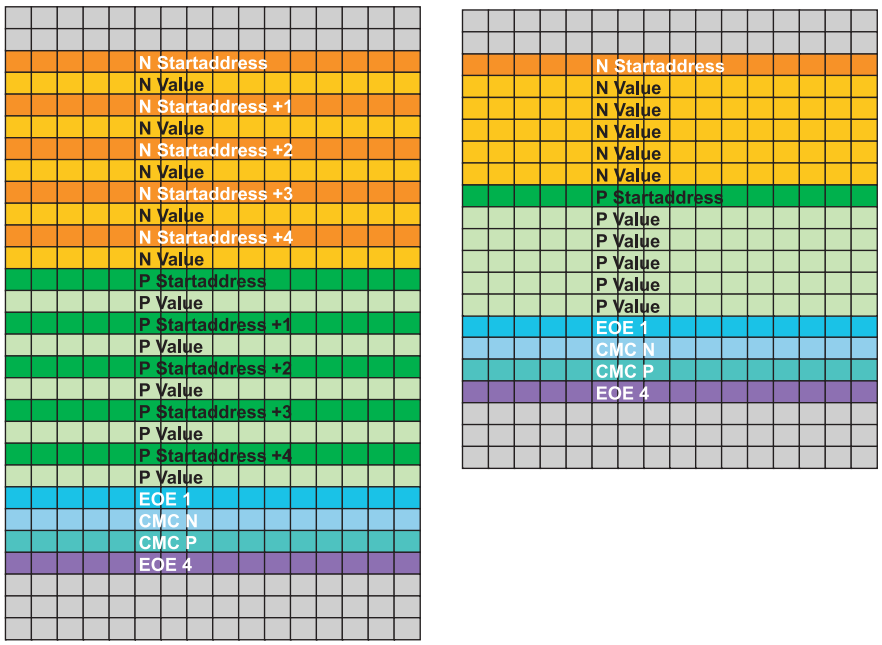


Figure 34: Comparison between compressed and non-compressed data output format. The event size is given as the number of FIFO words the event will occupy after preprocessing. The case shown here is the simplest case without cluster merging or stretching.

being the one in the middle. But due to charge sharing effects it is possible that two or more neighboring channels see enough signal to exceed their threshold. In this case, a larger cluster is stored, because the write gate is dynamically enlarged. The cluster now consists of 6 or more *cd*s with an address information in the selected format. In principle the only limit on the total cluster size is the detector size. If there is a block of n hit channels, the resulting cluster after ZS will have the size n+4, which is n plus 2 × cluster range, and include the two *cd*s before the first hit channel and the two *cd*s after the last hit channel. The cluster will, of course, not be expanded beyond the detector size (640 channels). In case there are two or more spatially separated hit channels in an event, e.g. generated by different particles, one cluster will be written for each hit channel. So the *hit multiplicity*[53] for an event corresponds to the number of clusters written. The only exception from this rule is the situation, in which there are two hits caused by different particles crossing that close together, that at least one channel within the scope of second cluster's cluster range lies within the cluster range of the

[53]The *hit multiplicity* in a module is not to be mistaken for the *track multiplicity* of the entire telescope system.

first cluster. The clusters overlap. In this case, a situation occurs which is called *cluster merging*: two hits appear as one large cluster if there are less than 2 × cluster range non-hit channels between the last hit channel of hit 1 and the first hit channel of hit 2. Situations like this (which are supposed to be rare) can be identified by having a closer look to the distribution of hit channels (i.e. hit bits) within a cluster for clusters larger than 7 channels.

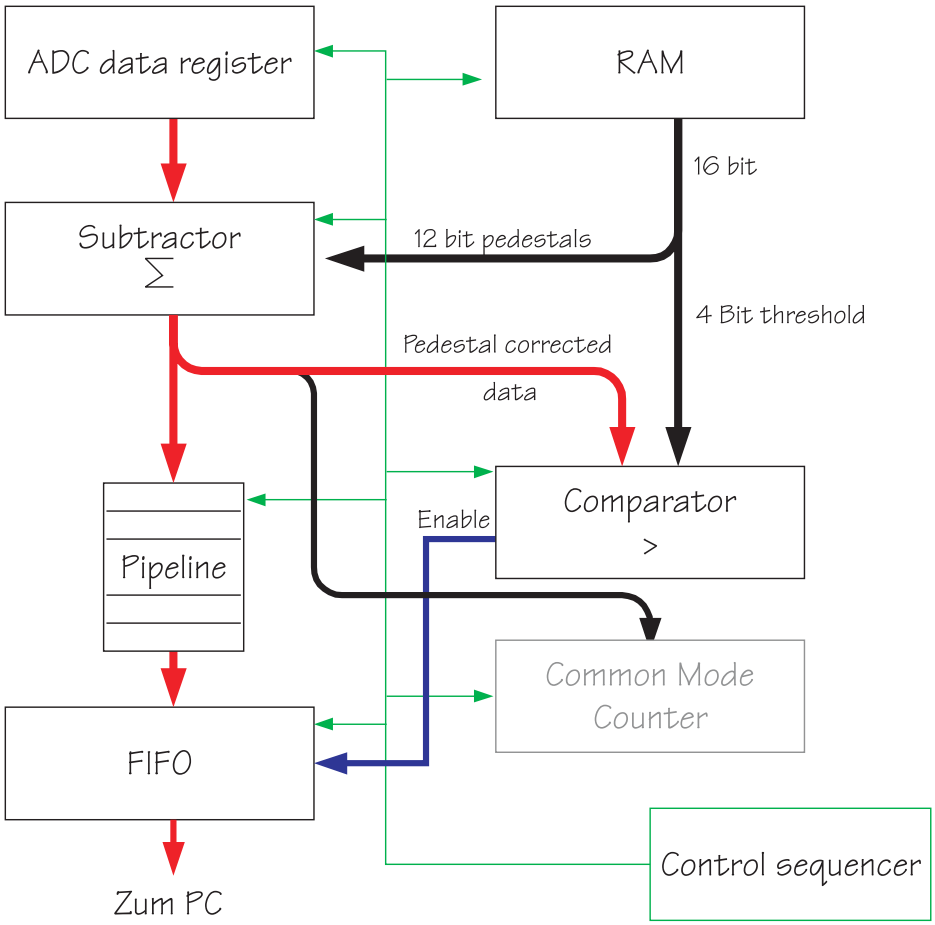


Figure 35: Flow chart of the data preprocessing. The data is received from the ADC data register (the demultiplexer). The subtractor performs the pedestal correction. The pedestal corrected data is fed into a pipeline shift register, while a comparator makes the "hit decision", which, if met, causes the cluster to be written into the FIFO. The common mode counter, basically a large accumulator, sums up all pedestal corrected ADC values. The whole procedure is controlled by the Control sequencer (which is the MSQ).

Implementation. A flow chart of the data preprocessing as implemented in the MFPGA is presented in fig. 35. Zero suppression and data compression are realized by steering the write gate signal, which controls the OMU. The OMU is operated by the MSQ. At the OMU the data from N- and P-compartment, which were processed independently and in parallel until then, are brought together and serialized again. The OMU applies data from N- and P- compartment at the FIFO input in the so called canonical order cycle *COC*, one of which is performed in each CCC :

1. N-compartment address information for channel n
2. N-compartment channel cd_n
3. P-compartment address information for channel n
4. P-compartment channel cd_n

Afterwards, the same order is repeated for channel number $n + 1$. A data word applied to the FIFO input is written into the FIFO if a *FIFO write enable* is applied for this data word. Whether a write enable is given or not is decided by the write gate. If e.g. channel n is hit, the write gate for channel n becomes active after c_{n+2}, when the HD section detected a hit. The hit bit is fed into the *gate generator* circuit which generates a write gate with the following behaviour (cluster write cycle):

- The write gate becomes first active when cd_{n-2} is available at the output of the pipeline shift register (3 CCCs after the hit bit entered the gate generator). For the N-side, the write gate will persist for step 1 and 2 of the OMU COC, for the P-side the write gate is active for step 3 and 4 of the COC.
- When cd_{n-1} is available at the output of the pipeline shift register, the write gate is active during
  - step 1 and step 2 of COC if DC is not active or
  - only step 2 of CO if DC is active

  In the first case address information for the N-compartment and *cd* value for the N-compartment are written, in the latter case only the *cd*-value for the N-compartment. Afterwards, write gate becomes active for
  - step 3 and step 4 of COC if DC is not active or

– only step 4 of COC if DC is active

In the first case address information for the P-compartment and *cd* value for the P-compartment are written, in the latter case only the *cd*-value for the P-compartment. This cycle repeats till the full cluster data is written.

- After processing cd_{n+2} (which is processed like cd_{n-1}) the write gate will become inactive and the gate generator will be reset. If a new hit word is detected, the cluster write cycle will start again.

- If a new hit word enters the gate generator before the cluster was fully processed, the cluster write cycle starts over but without writing a new first address information in compressed mode (DC active).

- No write gate will be generated for channel numbers < 1 or > 640. If a cluster exceeds channel number 640, it will be truncated. Care has to be taken if there are hits in channel 1 or 2. As a cluster belonging to these channels would actually start in channel -1 or 0, but for these channels the write of address information or data is vetoed and the cluster starts with a data word instead of an address information[54]. In principle, one can tell from hit bit and right (i.e. larger channel numbers) cluster edge, which channel had been hit even without having the address information, but most of the online and offline programs can't handle this case at the moment. A possibility to avoid this problem is to simply mask channel 1 and 2. They won't produce any hits then and the problem of writing non-existent strip addresses is avoided. Another possibility is to write cluster start address 0 instead of vetoing addresses. This solution requires some re- programming of the MFPGA (will be done in a later version).

As the output data is not sorted by compartment and moreover cluster data from different compartments can merge (e.g. if the same channel numbers on N- and P- side of detector are hit), the data is marked using the remaining 2 bits of the 4 LSBs which contain no useful information yet. This is necessary to enable the user to uniquely identify a word in FIFO. So it is possible to tell whether a given FIFO word is an address or a data word, and it can be told to which compartment (N or P) the data belongs. For more details about that please refer to the BAT output data format reference guide.

[54] This will only be a problem for compressed mode.

DPR operation modes For each DPR stage, except for the DD, of course, a separate register bit in CMD II decides whether this operation is to be applied to the data in the current session or not. Although the feature control bits can be set independently, not all combinations of features[55] make sense. Please note that all DPR control flags are "active low". Setting one of these bits will turn the corresponding feature off. These flags are:

1. *THRESHOLDS OFF*: CMD II, bit 0.
2. *SPARSE OFF*: CMD II, bit 1.
3. *PEDESTALS OFF*: CMD II, bit 2.
4. *COMPRESS OFF*: CMD II, bit 6.

In the past, only three DPR operation modes have been used:

- *Primitive.* In the primitive operation mode, all DPR features are turned off. In this way, the output data of the telescope consists of the raw, unprocessed output data of the ATC ADCs. Each channel data is escorted by an address word containing the channel's address. The RAM contents is not evaluated, so the RAM does not have to be initialized with proper values. This operation mode is used, together with the software trigger condition, mainly for pedestal acquisition and sometimes for debugging.
- *Hit detector.* In the hit detector operation mode, pedestal subtraction and hit detection are active. In this operation mode, the RAM values have to be initialized with reasonable data. Thus, pedestal and threshold values have to be prepared. The DPR is then generating a hit bit and adding it to the output data. But still all channels ADC values are written into FIFO together with their address information. This operation mode is used for system tests to see if the pedestal subtraction works, to check if the hit detection works (if an unmasked channel is always reported as being hit), to find noisy channels and channels for which a higher threshold might be more adequate (if they occasionally produce hits while only dummy triggers are given).
- *Full DPR.* In full DPR operation mode pedestal subtraction, hit detection, zero suppression and data compression are all active. This is the normal operation mode for testbeam data acquisition. The data size is at a minimum, because all redundant information as been left away.

[55] A combination of DPR features is called DPR operation mode.

Name	*PS*	*HD*	*ZS*	*DC*	*RAM contents*	*Application*
Primitive	no	no	no	no	not initialized	pedestal acquisition, debugging
Subtracted	yes	no	no	no	pedestals	unused
Hit detector	yes	yes	no	no	pedestals & thresholds	system tests, debugging, initialization
Uncompressed	yes	yes	yes	no	pedestals & thresholds	testbeam data acquisition (relaxed)
Full DPR	yes	yes	yes	yes	pedestals & thresholds	testbeam data acquisition (high performance)

Table 7: Overview about DPR operation modes. Only *Full DPR* and *Primitive* are used in the present version of the DAQ software.

The remaining operation modes either have less performance than one of the settings described here (e.g. operating with pedestal subtraction but without hit detection) or are not reasonable (e.g. operating with hit detection but without pedestal subtraction). Table 7 gives an overview about the different operation modes, boundary conditions and their application. Combinations not listed here are not to be used.

Address information In the previous description of DPR modes it has been discussed extensively under which circumstances an address information is written into the FIFO, but not, what the output format of the address information is. The address information consists of one 16 bit FIFO word. the 4 MSBs are, like the MSBs for a data word, used to uniquely identify an address word and the compartment, to which this address word belongs. One possibility to generate an address information would have been to implement a counter which counts the CCCs and to simply add this information on demand to the output data. For various reasons it was decided not to use an additional counter but to use the address information provided by the RAM address counter. As the RAM address counter has an individual value for each channel, the address information is unique. But as the counter counts at double speed (two addresses per CCC) and additionally has some offset, the address information has to be decoded to obtain the corresponding channel

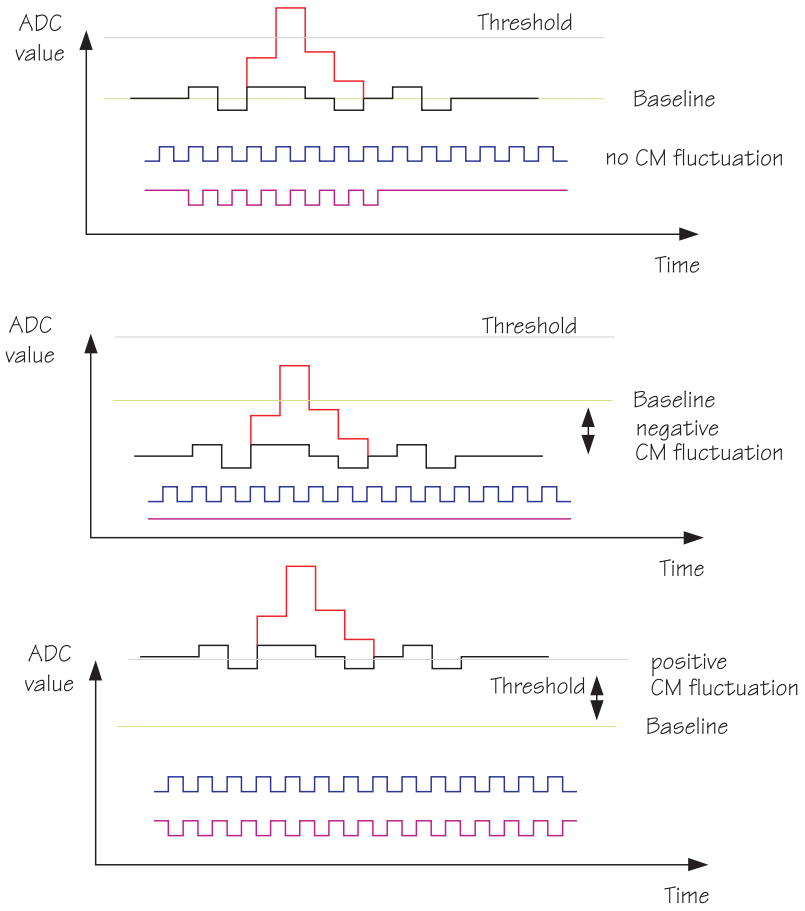


Figure 36: Common mode fluctuations. Negative fluctuations: Hit might get lost. Positive fluctuations: FIFO is flooded with useless data.

address. This can be done in the following way:

$$A = \frac{A_r - \Omega_c}{2}$$

with A being the decoded channel address, A_r the raw, coded channel address and Ω_c a compartment specific offset value, which is 21 for the N compartment and 22 for the P compartment.

Coping with common mode noise (CMN) Common mode noise is hazardous to the hit detection concept. Some of the problems caused by common mode fluctuations are shown in fig. 36. Therefore the design of the ATC and the system's settings have been optimized for low common mode noise. However, it is not possible to totally get rid of the common mode

noise. Moreover, for an accurate analysis of pulse height correlations and pulse height distributions (which are possible due to the analog readout) the common mode contribution to the signal amplitudes in this event is to be subtracted[56]. There is also another benefit in observing the common mode values. In the described operation scenario pedestals are only taken once at the beginning of each run. If, for some reason, the pedestals should start to drift in some direction[57], one could easily notice this by looking at the development of the common mode. In case such a drift is observed, one can simply take new pedestals[58], or, if it is due to some problem, fix the problem before the start of a new run. In common telescope architectures, in which all output channels provide an output signal, this can be done offline by simply calculating the mean value of all output signals for one event while leaving out the hit cluster(s). Due to the on-module zero suppression feature provided by the BAT, this calculation cannot be done offline (at least if the telescope is operated with ZS active). The implementation of a full common mode correction circuit would have exceeded the MFPGAs resources, so a "hybrid" solution, half online, half offline, has been realized. The DPR provides a value with which the offline common mode correction is possible even with ZS active.
The common mode correction value *cmc* for signal amplitudes in offline analysis is calculated in the following way:

$$cmc = \frac{\sum_{i=1}^{N_{max}} cd_i - \sum_{j=1}^{NC} \left(\sum_{k=0}^{CCN_j - 1} cd_{\left(n_{start_j}+k\right)} \right)}{N_{max} - \sum_{m=1}^{NC} CCN_m} \qquad (1)$$

N_{max} is the total number of channels read out, NC is the total number of clusters detected in this event, and CCN is the "channel count", the number of channels inside the cluster. n_{start_j} in this context is the start address for cluster number j. The first term in the numerator is simply the sum of all *cd* values, the "event amplitude". The second term in the numerator, the double sum, is the "cluster amplitude", that part of the event amplitude which is contributed by the hit cluster(s)[59]. The term in the denominator is simply the number of channels which contribute to the cluster amplitude.

[56] This also improves the signal to noise ratio.

[57] This could happen due to supply voltage or temperature drift.

[58] However, one will have to finish the old run; pedestal acquisition during a run is not supported.

[59] The most desirable case is, for various reasons, the case that there is only one hit cluster in an event.

When the BAT is used for data acquisition with active ZS, the part of equation 1 which can not be calculated from the data read back for an event is the event amplitude, because most of the cannels' *cd* values are simply discarded. Therefore the DPR provides exactly this value.
A building block called "common mode counter"[60] sums all *cd* values for all channels for each event independent from their status as hit channels, cluster channels or irrelevant channels. This data is transmitted to the DAQ with the end of event data (EOED, see the section about SSM). The value obtained this way is called "common mode count" *comco*. With the *comco*, the *cmc* can be calculated in the same way as in equation 1:

$$cmc \;=\; \frac{comco - \sum_{j=1}^{NC}\left(\sum_{k=0}^{CCN_j-1} cd_{\left(n_{start_j}+k\right)}\right)}{N_{max} \;-\; \sum_{m=1}^{NC} CCN_m} \tag{2}$$

using the same definitions as given for equation 1. All the other values needed can easily be calculated form the output data acquired with ZS active. The *cmc* corrected cluster amplitude for example, can be calculated in the following way:

$$ca_j \;=\; \sum_{k=0}^{CCN_j-1} cd_{\left(n_{start_j}+k\right)} \;\; - \;\; cmc \;\times\; CCN_j$$

As each detector side has its own common mode fluctuations which can be very different[61], each ATC compartment has its own common mode counter, which is acting independently. Although some effort was made to keep the common mode noise as low as possible, the common mode counters were designed to cope with large common mode values. The common mode counters are 20 bit counters, so for 640 channels a maximum common mode fluctuation of about 400 ADC counts in either direction (positive and negative) for each channel can be handled, before an overflow occurs.
It turned out that in spite of the fact that by design and biasing common mode is supposed to be low, the actual common mode behaviour is strongly dependent of the experimental environment the system is to be operated in. At CERN in the H8 testbeam environment for example the common mode contributions to the systems noise was much larger than those observed at the E3 testbeam facility at the ELSA accelerator in Bonn. It turned out that

[60] This is basically a huge accumulator.

[61] Although common mode noise turned out to be strongly correlated between N- and P- side of the detector, the fluctuations can differ in size.

connecting the system's power supplies via a separation transformer to the power lines is an easy and efficient way to reduce common mode power line pickup to the smallest value possible under the given conditions.

7.3.9 The shutdown state machine (SSM) and the interrupt control (IRQC)

The shutdown state machine waits for the MSQ signaling that one event has been fully processed. This means, that the analog data of all detector channels has been read out, preprocessed and, where appropriate, written to FIFO. If the MSQ signals such an occurrence, the SSM does the following things:

- The EOED is written into the FIFO,
- The IRQC block is alerted to check, if an IRQ condition was met while processing the previous event,
- The trigger number counter, which counts the number of fully processed events, is incremented and
- After finishing all this stuff, the SSM signals the MSQ that the shutdown was completed[62].

The EOED consists, depending on the DPR settings, of 4 to 5 words. If DC is not active, the EOED consists always of 4 words:

1. *EOE1: Trigger word.* EOE1 is a 16 bit word, like all FIFO data words are. The 4 MSBs contain a ID which allows a DAQ task to uniquely identify this word as an EOE1. The 12 LSBs contain the 12 bit trigger number as counted by the trigger number counter. This allows the DAQ tasks to keep track of the trigger numbers and to check if there are events missing.
2. *EOE2: CMCN I.* EOE2 contains the 16 LSBs of the 20 bit CMC word counted by the N compartment's common mode counter for this event.
3. *EOE3: CMCP I.* EOE3 contains the 16 LSBs of the 20 bit CMC word counted by the P compartment's common mode counter for this event.

[62] The MSQ needs this information to become idle again

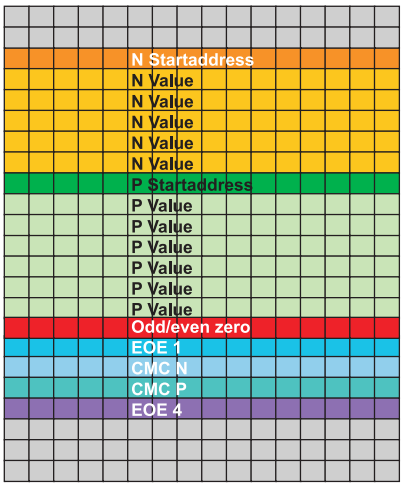


Figure 37: Example for an event (in FIFO) with an odd-even zero. Again, the number of data words in FIFO is given. The event given is the second simplest case, where only one of the two event clusters is 6 instead of 5 strips large.

4. *EOE4: CMCN & P II.* The 8 MSB's of EOE4 contain another header used to uniquely identify this word as EOE4. The 8 LSB's contain the MSB's of CMCN and CMCP (which are actually 20 bit values). The bits 0 to 3 contain the bits 16 to 19 of CMCN, the bits 4 to 7 contain the bits 16 to 19 of CMCP (in this order).

If DC is active, care has to be taken because then the condition might occur that an event written into the FIFO consists of an odd number of FIFO words. As will be explained in the section about FIFO control, this condition must't occur. Before writing the EOE 1 to 4 the shutdown state machine looks at the odd/even flag, which indicates whether an odd or an even number of words has been written into the FIFO for this event[63]. If the number was odd, one additional FIFO word, the so-called "Odd-Even Zero" (OEZ) will be written into the FIFO before writing the EOED. This special word contains no useful information. Its 4 MSBs allow once again a unique identification of this word as OEZ, and the rest are all zeros. An example for an event which has an odd-even zero is shown in fig. 37 More details about EOED and its decoding can be found in the BAT output data format reference guide.

The interrupt control (IRQC) stage works closely together with the SSM. The IRQC's task is to generate an IRQ if an IRQ condition is met. There are two different kinds of IRQ operation:

[63] This is independent from the DC setting: in uncompressed mode, the number of FIFO words will always be even.

DIRQ	Bit settings	Data words	Event number	≈IRQ rate
Rate 1	00	1024	64	16 Hz
Rate 2	10	4096	256	4 Hz
Rate 3	10	16384	1024	1 Hz
Rate 4	11	32768	2048	500 mHz

Table 8: Overiew over the different IRQ rates. The number of events is given for "minimal clusters" and the IRQ rates are given for a trigger rate of 1 kHz.

- *Event interrupt:* An event interrupt (EIRQ) is generated, if the module is in event interrupt mode (CMD II bit 3 is not set). If in this mode a trigger occurs which is processed by the MSQ, the IRQ will be signaled on IRQ channel 1 as soon as the event is fully processed. An event is fully processed as soon as the EOED has been written by the SSM. The IRQ will persist as long as the FIFO empty flag is inactive, indicating that there is still some data in the FIFO. If the FIFO is fully read out, the IRQ will vanish.

- *Data interrupt:* A data interrupt (DIRQ) is generated if the module is operated in data interrupt mode (CMD II bit 3 is set). A DIRQ will be signaled as soon as the data interrupt condition (DIRQC) is met and the SSM finished processing the event during the processing of which the DIRQC was met. The DIRQC is met if the number of data words, which was written into the FIFO since the begin of run or since the last DIRQ occurred, exceeds a certain number, the DIRQ threshold (DIRQT). There are 4 different settings of the DIRQT, which can be selected via the IRQ-Rate bits (CMD II bits 4, 5)[64]. Table 8 gives an overview over the different settings. The event numbers given are calculated for "minimal cluster" events (5 *cd*s, 1 hit channel per cluster with DC active). The IRQ rate is calculated for 1 kHz trigger rate. It is inversely proportional to the trigger rate. The numbers may change for different DPR settings and data quality[65]. The DIRQ will persist until the IRQ is reset by the DAQ software by activating MA 25 (RESIRQ, a strobe address).

In spite of the fact that, in EIRQ mode the interrupt does not have to be reset by RESIRQ, it is necessary to do a RESIRQ. This is because the ARM-trigger signal (see section about triggering the MSQ) is disabled after a trigger

[64]The name arises from the fact, that the rate of IRQs on the BB varies with different IRQ rate bit settings, depending on the trigger rate.

[65]Data quality is used here as an umbrella term for event multiplicity, cluster sizes etc..

accept and has to be enabled again, which can be done with RESIRQ.
For both EIRQ and DIRQ the IRQ is caused by a certain amount of data written into the FIFO. This data of course has to be read back by the DAQ task which is alerted by the IRQ. The DAQ task is set up to read back full events. For the EIRQ things are easy. As the EIRQ will persist until the FIFO is empty and by the ARM-trigger signal further trigger accepts are vetoed, one can be sure that only one event has been written into the FIFO. One then can read exactly one event. The FIFO should be empty then, and with an RESIRQ, which resets the ARM-trigger, the cycle can begin once again.
The EIRQ processing cycle works like this:

1. EIRQ is sent after first event (IRQ condition is met if FIFO is not empty).
2. IRQ is detected by the DAQ task.
3. DAQ task reads back one event.
4. After finishing readback, RESIRQ is given to re-activate ARM-trigger.

However, for the DIRQ things are not that easy. In DIRQ mode, the trigger is not disabled if an IRQ is sent. As the events may have different sizes, the number of events which caused the DIRQ may differ, and during the time the DAQ needs to process the IRQ additional events may be processed, the total number of events stored in the FIFO is unknown and, what is worse, even may change while the DAQ is reading out some data.
For the DAQ has to know which number of events has to be read out for a certain IRQ, the so-called data size counter (DSC) counts the number of data words which have been written between the time of the last DIRQ and the time the new DIRQ is signaled. This counter can be read back by reading MA 7 (low byte) and MA 24 (high byte). As the DIRQ is only signaled if an event is fully processed (including EOED), one can be sure that the number of data words written corresponds to a number of complete events (this is important for the writer and the monitoring tasks). So one can read back on event-by-event basis until the number of data words read corresponds to the value of data words which have been written, according to the value indicated by the DSC. One might be confused here why the number of data words and not the number of events is transferred to the DAQ task, as the readback is done on event-by-event basis. This is because the number of data words is important for monitoring the FIFO fill level and the event size may differ. Moreover, also counting the number of events since the last DIRQ would

have required another counter in a timing-critical section of the FPGA, and resources are low.
After finishing one event readback, a RESIRQ has to be given to clear the IRQ like in DIRQ mode [66]. The DIRQ processing cycle works like this:

1. DIRQ is sent after detecting enough data to fulfill the IRQ condition.
2. IRQ is detected by the DAQ task.
3. DAQ task reads back both bytes of the DSC.
4. DAQ task reads back event by event, comparing data size read with data size from DSC.
5. After finishing readback, RESIRQ is given to re-activate ARM-trigger.

Please note that this processing cycle does not really differ from the processing cycle for the EIRQ, although it may appear so. For the EIRQ "enough data" simply means one data word, and the DSC works for both modes in exactly the same way, so reading back the DSC in EIRQ mode will simply give the event size of the single event stored in the FIFO. Thus, the DAQ task does not even have to know which IRQ mode is set. The same processing cycle works for both modes.

Emergency busy & multiple IRQs. It is probable that during the time a pending DIRQ was waiting for being and being processed, enough data gathered in the FIFO to signal the next DIRQ. This IRQ will be signaled then immediately after the RESIRQ of the preceding DIRQ. The number of data words which is stored in the DSC register then is not the number of data words by the time the event for which the IRQT was exceeded was fully processed, as one might suspect. Of course, this would be feasible. But especially at low IRQTs and high data rates it would require a complex system of IRQ and DSC value pipelining. To avoid this the number of data words which is stored in the DSC corresponds to the number of data words which is stored in the FIFO by the time the IRQ is *signaled* (that is right after finishing with the preceding IRQ). In this way, neither IRQs nor data will be lost, and the DAQ task has an easier job, for it can get with one processing cycle an amount of data which, under different circumstances, would have

[66] Please note that although the reasons why the RESIRQ has to be given are different, it has to be given in both IRQ modes.

caused several IRQs[67].
But if an IRQ is pending for quite a long time without being processed and the trigger rate is very high, the FIFO might become full. This must not happen, because data written in the FIFO while it is full will definitely be lost. In this condition one might lose a part of the last event and, even worse, entire events following the last event that has been partially written into the FIFO.
To prevent this weird condition, the *emergency busy flag* (EBF) has been introduced. In principle this is an additional IRQT, which signals an IRQ if more than a certain number of data words has been written into the FIFO. But in addition to generating an IRQ, which should have been set by IRQC yet in most of these cases anyway, the status of this flag is also taken into account while generating the MODULE-BUSY. A module, whose EBF is active will send a MODULE-BUSY as long as the EBF is active, thus preventing further trigger accepts coming from the TLU and bringing the whole telescope system to a halt[68], giving the DAQ task enough time to process all pending IRQs, including the IRQ sent by the module which stopped the system with its EBF.
It turned out that using the FIFOs *programmable almost full flag* (PAFF) for this task is very convenient. The PAFF can be set to any value beyond 32k. It is referred to the real FIFO fill level (not to the "mirror image" counted inside the FPGA). The PAFF will become inactive again, if the FIFO fill level drops below the number of events programmed in the PAFF register inside the FIFO. Care should be taken to set the PAFF to a level, at which an event being processed when the EBF condition is just being met can still be fully written into the FIFO and there is enough space for even one more event[69]. During the last testbeam periods the BAT has been used in, the PAFF was set to a level of 1000, leaving a space of 1000 data words between the top end of the FIFO and the point where the trigger was turned off. This correspond to 62 minimum size events; 1 and $\frac{1}{2}$ 640 strip events in DC mode (weird!) would also fit in.

Acquisition mode & sequencer runtime. In DIRQ mode the only limit on the event rate is given by the sequencer runtime as described in the section about the MSQ. As long as no EBF turns off the trigger, the maximum

[67] Data readback is not that time consuming than identifying different IRQ sources. For this reason, the IRQ rate should be kept as low as possible for a given trigger rate.

[68] This is necessary to enforce trigger synchronity.

[69] This is because the TLU takes some time to turn off the trigger as a reaction to the active MODULE-BUSY, and during this time one more trigger might slip by.

event rate is 7.4 kHz, corresponding to the reciprocal value of the sequencer runtime. To fully exploit this high data rate capability it is necessary to

1. Operate at high IRQTs. Thus, the system will have fewer IRQs with a lot of data instead of lots of IRQs with few data. This is more time efficient.

2. Operate at low VA 2- shaping times, not because a low hold delay will decrease the sequencer runtime, but because the data quality will be higher. Especially the event multiplicity will be one for a higher fraction of events. As will be explained in the section about front-end electronics, particles, whose trigger coincidence was suppressed by the TLU due to the MODULE-BUSY coincidence will also generate a signal in the VAs. If the signal is not fully decayed when the next trigger occurs, it is not unlikely to produce an additional hit cluster in the DPR [70].

The EIRQ mode was originally developed for low event rate operation. As one EIRQ has to be processed for each event and each module, the system will slow down simply because of the large number of IRQs. Moreover, the system's dead time in EIRQ mode is made up of the sequencer runtime and the *readout time*, the time it takes for the DAQ to read back the data from the module. And this readout time has to be multiplied by the number of modules connected, because the MSQs run in parallel, but the modules are read out serially. So operation in EIRQ mode is not very recommendable if one wants to achieve high event rates.
However, integrating DUTs in the DAQ readout chain can be done in the easiest way by doing an event by event readback of telescope and DUT data. It is the only way of integrating DUTs which do not provide data buffering and IRQ support[71], and it is a way of testing DUTs in which data buffering is not yet implemented[72]. The data rate can be increased by running in a "hybrid" IRQ mode, where one module generates an IRQ for every event and the other modules run in buffered mode, so the readout time only multiplies by the number of the IRQ generating modules (can be 1) and the number of DUTs. In a later version of the DAQ software, the TLU will take the task of generating IRQs for EIRQ mode (more on this in the section about the TLU).

[70] In most cases this hit has a much lower amplitude than the real, the *in-time* hit, so the correct hit can be identified using the pulse height correlation.

[71] This is a problem for e.g. VME-based DUTs.

[72] Every DUT, which uses the DTC as an interface to the BB, is able to perform both data buffering and IRQ support.

7.3.10 The FIFO control (FICO)

The FIFO control block (FICO) is responsible for correctly writing data to and reading data from the FIFO. The FIFO data is organized in 16 bit words. Writing of FIFO data is either done automatically during DAQ (controlled by MSQ or SSM) or can be done manually for test purposes. To write the FIFO manually, the TESTMODE (CMD I, bit 7) has to be active. As the FIFO is 16 bit wide, a 16 bit word has to be prepared in the FIFOLO and FIFOHI registers in FICO, 2 8-bit registers which can be accessed using the MA 17 (RWFIFODATLO) and MA 18 (RWFIFODATHI). FIFOLO and FIFOHI represent the high or low byte respectively of the data which is to be written in the FIFO. After preparing the FIFO word, the data is transferred into the FIFO by accessing the RW-FIFO register (MA 16) and writing a dummy value (e.g. 0). There is no register connected to MA 16, the write access only generates a FIFO-Write-Enable which simultaneously writes both bytes of the data word prepared in the FIFOLO and FIFOHI registers into the FIFO. The FIFOLO and FIFOHI registers are also used for programming of the FIFO's programmable flags.
The only way to read FIFO data out of the FIFO is by using the *read DWORD* access. During a DWORD access a 32 bit wide data word is read from the FIFO. The data is transferred in 4 packages at 8 bit each. This access is very fast. To avoid timing problems between the FIFO and the IFPGA, the main XILINX provides the so-called FIFO-DWORD (FICO) register. This is a 32 bit wide register, in which a 32 bit word consisting of 2 16-bit FIFO words is prepared. If a DWORD access occurs, the 4 bytes are read out of the FIFO-DWORD register and transferred to the PC. After the DWORD access has been finished, the FIFO-DWORD register autonomously takes another 2 FIFO words out of the FIFO. So the next DWORD access is prepared. The circuit section which controls the withdrawal of a new 32 bit word out of the FIFO is a state machine (FIFOCON) written in verilog HDL, as well as the circuit which steers the data output during a DWORD access. As there are 2 FIFO words read out during one DWORD access, the order in which they are read is important. If the words to be read are the words number n and n+1, the order is:

1. FIFO word n lo byte
2. FIFO word n hi byte
3. FIFO word $n + 1$ lo byte
4. FIFO word $n + 1$ hi byte

MSBs			LSBs
Bits 31-24	Bits 23-16	Bits 15-8	Bits 7-0
Word $n+1$ hi byte	Word n+1 lo byte	Word n hi byte	Word n lo byte

Table 9: Structure of a 32 bit data word obtained by a DWORD access to FIFO control during normal data acquisition.

Thus, the 32 bit data word read back will look like indicated in table 9. This process only works if there are at least two words of unread data in the FIFO. If not, the FIFOCON would try to access an empty FIFO. For accessing an empty FIFO would result in getting senseless data (two times the last word written in FIFO), the FIFOCON has to know if there are only one or no words left in the FIFO. This is done by checking the programmable FIFO almost empty flag (PAEF). The PAEF can be set to any value between 0 and 32 k and becomes active if an equal number of data words or less than the programmed value is left in the FIFO. The FIFOCON does not access the FIFO if the PAEF is active, so the value the PAEF is to be programmed with is 1.
There are two ways to find out if the FICO has valid data for a FIFO access in its registers when in doubt. First, checking the FIFO empty flag or the PAEF. If one word is left in FIFO, the FIFO empty flag is inactive, but the PAEF is active, if the FIFO is empty, both flags are active. But only looking at the FIFO flags is not enough, as the FIFOCON may have read the FIFO empty, but the data stored in the FICO register has not been transferred to the PC yet. This condition is indicated by the "OLD" flag, which can be read using the status bus (MA 10, bit 6). The OLD flag is generated by the FIFOCON and indicates, that a value stored in the FICO registers has been transferred to the PC. The OLD becomes active after a valid DWORD access and remains active until a new valid 32 bit word consisting of data taken from the FIFO is stored in the FICO registers. So in normal operation, the old flag becomes active after the DWORD access and becomes inactive again as soon as the FIFOCON finished its access to the FIFO (about 100 ns later, but in any case sooner than the next DWORD access).
So, the FIFO has been fully read out if the FIFO empty flag and the old flag are both active. If the FIFO empty flag is active but the old flag is not, there is exactly one word of the former FIFO contents left, which has not been read out. This is then stored in the FICO registers[73].
A deadlock occurs for this method, if there is an odd number of words writ-

[73] One can think of the FICO registers as an 2 word extension of the FIFO, so the FIFO is not 64k but 64k+2 words long.

ten into the FIFO. If the DAQ software tries to read the FIFO empty, the situation occurs, that at some time there might be only one word left in the FIFO as there are always 2 words read by the FIFOCON. In this case the PAEF becomes active, thus preventing the FIFOCON to read the remaining word from the FIFO. So there is no way to get this remaining word out of the FIFO as access to the FIFO data is only possible trough the FICO registers. In this case, the OLD flag will become and stay active, although the FIFO is not empty (but the PAEF is inactive). This is a catch 22 which is to be avoided under all circumstances.
The safest way to avoid this is to simply ensure that for each event only an even number of FIFO words is written. In the present MFPGA configuration this is ensured by the SSM, which writes the OEZ, if the number of words written is odd[74].

7.3.11 The IO blocks (IOB)

The MFPGA IO can be split in three groups:

1. IFPGA communications
2. RAM / FIFO IO
3. ATC I/O

IFPGA communications In the IFPGA communications all signals are included which are passed between IFPGA and MFPGA in either direction. Pin assignments for the MFPGA and a short explanation of the signal purpose are given. The signals are listed in table 10. Pin # is the pin number of the MFPGA. For the sake of completeness the LEMO signals are also listed, although they are not used in the IFPGA, but directly transmitted to the module's LEMO connectors. There are only three of them, and they serve very special purposes:

1. *LEMO 1*: Module busy. This is most important for coordinated telescope operation. The module-busy signal is evaluated in the TLU, in which a coincidence of all connected module's *module busy* decides whether a trigger can be given or not. A module is "busy" if

[74] This is only a problem of Full DPR acquisition mode, because for all other acquisition modes there are only even numbers written anyway.

Name	Pin #	Description
DOUT 0	137	Data output
DOUT 1	140	
DOUT 2	135	
DOUT 3	122	
DOUT 4	127	
DOUT 5	121	
DOUT 6	115	
DOUT 7	116	
DIN 0	134	Data input
DIN 1	117	
DIN 2	138	
DIN 3	146	
DIN 4	128	
DIN 5	147	
DIN 6	144	
DIN 7	146	
DWORD	129	Indicates dword access
RD-CLK	132	Read clock for read accesses
READ / $\overline{WRITE}$	145	Indicates a read access
MODE / $\overline{DATA}$	124	indicates a mode access
$\overline{WRITE\ \ ENABLE}$	141	active low write enable for write accesses
MODE-ADDRESS 0	143	Mode address bus (from IFPGA)
MODE-ADDRESS 1	126	
MODE-ADDRESS 2	125	
MODE-ADDRESS 3	139	
MODE-ADDRESS 4	110	
IRQ-ADDRESS 0	162	IRQ address bus (to IFPGA)
IRQ ADDRESS 1	152	
LEMO 1	120	Module busy output
LEMO 2	118	Inhibit input
LEMO 3	123	External trigger input

Table 10: IFPGA communications. Pin assignments.

(a) If the module is not yet fully configured (CMD I bits 2,3 are not set to external trigger mode[75]),

(b) In DIRQ mode if the EBF is active or the MSQ is running or

(c) In EIRQ mode if the MSQ is running or if the module's FIFO has not been fully read out yet.

2. *LEMO 2*: Inhibit. This is a signal which is no longer used, as the trigger control has been passed to the TLU in the meantime. If an inhibit signal is applied, the external trigger signals are blocked and the module won't accept external triggers as long as the inhibit signal persists.

3. *LEMO 3*: Trigger. If a TTL pulse is applied to this LEMO input and the module is configured properly, the MSQ will start data taking.

RAM / FIFO IO The signals listed here are used for the communication between RAM or FIFO and the MFPGA. Normally, there are lots of data signals and few control signals. For the FIFO there are 2 16 bit data busses (one for writing into and one for reading out data from the FIFO), for the RAM only one 16 bit data bus and one 16 bit address bus. As most of the signals need synchronous timing, they are synchronized at the output pads using a FPGA output flip-flop. The pin assignment for the FIFO is given in the tables 11 and 12, the pin assignment for the RAM is given in the tables 13 and 14. Pin # is the pin number of the MFPGA. For a detailed description of the component features or the functionality of these control signals, please refer to the corresponding sections in this manual or the component datasheets.

ATC I/O The ATC I/O includes all control and data signals going to / returning from the ATC compartments. Each of these signals passes through one channel of the coupler gate. For a detailed description of the signals please refer to the ATC section. The signals for the N-compartment are listed in table 15, the signals for the P-compartment are listed in table 16. Channel # is the number of the coupler gate channel as given in the DTC schematic. Port # is the pin number on the ATC port, the connector between ATC and DTC. Pin # is the pin number of the MFPGA. An asterisk (∗) indicates that this signal is only needed for VA hybrid operation, a dagger (†) indicates that the signal is needed for both, VA and ATC component

[75] Because this is the last thing which is done in a module's *prepare for readout* routine.

Name	Pin #	Description
FIFO data out 0	40	Output data from FIFO
FIFO data out 1	59	
FIFO data out 2	61	
FIFO data out 3	64	
FIFO data out 4	66	
FIFO data out 5	69	
FIFO data out 6	71	
FIFO data out 7	73	
FIFO data out 8	75	
FIFO data out 9	80	
FIFO data out 10	81	
FIFO data out 11	83	
FIFO data out 12	84	
FIFO data out 13	86	
FIFO data out 14	88	
FIFO data out 15	91	
FIFO data in 0	60	FIFO data input
FIFO data in 1	63	
FIFO data in 2	65	
FIFO data in 3	68	
FIFO data in 4	70	
FIFO data in 5	72	
FIFO data in 6	74	
FIFO data in 7	76	
FIFO data in 8	82	
FIFO data in 9	85	
FIFO data in 10	87	
FIFO data in 11	89	
FIFO data in 12	92	
FIFO data in 13	94	
FIFO data in 14	96	
FIFO data in 15	98	

Table 11: Pin assignment for FIFO part I: Data busses.

Name	Pin #	Description
FIFO empty flag	114	FIFO is empty
FIFO almost empty	46	equal or less data words than programmed for the PAEF
FIFO almost full	42	equal or more data words than programmed for the PAFF
FIFO full flag	41	no write space left in FIFO
FIFO LOAD	112	Program PAFF / PAEF values
FIFO RESET	113	Reset FIFO.
FIFO WRITE ENABLE	43	Writes data in FIFO
FIFO READ ENABLE	111	Takes data from FIFO

Table 12: Pin assignment for FIFO part II: Control signals.

operation. Please note that all ATC output signals can be monitored by taking the signal from the connector's row right between the MFPGA and the coupler's gate. Here, the order of the signals is important. Holding the DTC card with the coupler's gate down, the enumeration follows from the left to the right the following scheme (starting from left to right, with an $*$ indicating a not-connected pin or a missing connector):

28	26	25	27	$*$	24	...	11	$*$	8	6	5	7	$*$	4	2	1	3

7.4 The module's FIFO

In this section the FIFO's features are described as far as they are important for the normal telescope operation. For more information about the FIFO's features please refer to the datasheet. The module's FIFO is a Cypress type CY 4285-15JC in a 68 pin PLCC ceramic case. The FIFO is set up for asynchronous read and write. The FIFO is 18 bit wide (16 of which are used in telescope operation) and 64k deep. Although the FIFO can be written by software, its major task is to buffer the data taken by the MSQ and SSM during testbeam data acquisition and, in cooperation with the FICO stage in MFPGA, to make the data available fast in case the DAQ processes start a readback.
Writing to and reading from the FIFO is controlled by the interaction of two

Name	Pin #	Description
RAM address 0	188	Output pins for RAM
RAM address 1	187	address bus
RAM address 2	186	
RAM address 3	185	
RAM address 4	184	
RAM address 5	163	
RAM address 6	164	
RAM address 7	165	
RAM address 8	178	
RAM address 9	179	
RAM address 10	180	
RAM address 11	181	
RAM address 12	203	
RAM address 13	202	
RAM address 14	201	
RAM address 15	200	
RAM data in / out 0	190	I/O pins for RAM data
RAM data in / out 1	191	
RAM data in / out 2	195	
RAM data in / out 3	197	
RAM data in / out 4	198	
RAM data in / out 5	196	
RAM data in / out 6	193	
RAM data in / out 7	192	
RAM data in / out 8	177	
RAM data in / out 9	176	
RAM data in / out 10	175	
RAM data in / out 11	174	
RAM data in / out 12	173	
RAM data in / out 13	172	
RAM data in / out 14	170	
RAM data in / out 15	169	

Table 13: Pin assignment for RAM part I: Data / address bus.

Name	Pin #	Description
RAM WR-SEL LO	167	Selects lo byte to write
RAM WR-SEL HI	168	Selects hi byte to write
$\overline{RAM \quad OE}$	166	Selects write or read mode (output only active in read mode)
RAM WE	199	Writes data into RAM cell selected by address

Table 14: Pin assignment for RAM part II: Control signals.

Name	Pin #	Channel #	Port #	Description
CTR1 N	5	28	45	Select clock mode / CS
DRT N	6	26	46	Digital reset$*$
TON N	7	25	41	Test on $*$
CTR0 N	10	27	42	Select clock mode / CS
HLD N	11	24	39	Hold $*$
DTI N	12	22	40	Data in †
STR N	13	21	35	Strobe
CLK N	15	23	36	Clock †
S1 N	18	18	34	Multiplexer
S0 N	19	17	29	select bits
MN 22	16	20	33	ATC data output
MN 21	20	19	30	bits displayed here
MN 12	24	15	28	are depending on
MN 11	21	16	27	S0, S1 settings

Table 15: ATC I/O pin assignment part I: N-compartment.

Name	Pin #	Channel #	Port #	Description
CTR1 P	28	10	22	Select clock mode / CS
DRT P	23	13	23	Digital reset∗
TON P	22	14	24	Test on ∗
CTR2 P	29	9	17	Select clock mode / CS
HLD P	32	6	16	Hold ∗
DTI P	27	12	21	Data in †
STR P	30	11	18	Strobe
CLK P	33	5	11	Clock †
S1 P	31	8	15	Multiplexer
S0 P	34	7	12	select bits
MP 22	38	1	5	ATC data output
MP 21	36	2	10	bits displayed here
MP 12	39	3	6	are depending on
MP 11	35	4	9	S0, S1 settings

Table 16: ATC I/O pin assignment part II: P-compartment.

signals each: the write clock (WCLK) and the write enable (WE) for the writing, the read clock (RCLK) and the read enable (RE) for the reading of data. A data word is written, if the WE is active by the time a rising edge of the WCLK occurs. A data word is written, if the RE is active by the time a falling edge of the RCLK occurs. WE and RE are both active low. Please note that due to the design of the FIFO output stage of the MFPGA all FIFO signals are active high inside the MFPGA, independent of their polarity at the FIFO pin. According to the datasheet, RCLK and WCLK can be independent clocks at different frequencies. However, while testing the device it turned out that operation of the device with satisfying performance with respect to timing is only possible if

- both RCLK and WCLK operate with the same frequency and the access timing is controlled by WE and RE and
- both FIFO clock pins are not driven by a MFPGA pin, because this can result in design-dependent clock skew worsening the timing performance and preventing satisfying device operation.

In the present version of the DTC, the connection of RCLK and WCLK was originally made to MFPGA pins. This had to be changed by disconnecting these pins on MFPGA side and connecting the pins directly to the module's crystal oscillator by small wrap wires, thus forcing fixed timing between

MFPGA clock and FIFO RCLK and WCLK[76].
The FIFO possesses 4 flags, which inform the user about the FIFO's fill level. All flags are active low at the FIFO pin and active high inside the MFPGA. The available flags are:

1. The FIFO empty flag (EF): The FIFO empty flag indicates that there is no data left in the FIFO. The EF becomes active a short time after the rising edge of the RCLK with which the last word is read out of the FIFO. In telescope operation the EF is used to control the access of the FICO to the FIFO and to generate EIRQs. See the section about FICO and IRQC for details.

2. The FIFO full flag (FF): The FIFO full flag indicates that there is no write space left in the FIFO. The FF becomes active a short time after the rising edge of the WCLK with which a word is written to the last free FIFO cell. The FF is not used in normal telescope operation. If the FF becomes active in normal telescope operation, something went terribly wrong.

3. The programmable FIFO almost empty flag (PAEF): The PAEF can be set by the user to any value between 0 and 32k. The value is programmed by giving a number between 0 and 32k. The PAEF becomes active if there are equal or less words than the programmed value stored in the FIFO. For telescope operation, the PAEF is used to control the access of the FICO to the FIFO (see the section about FICO for details). For this purpose, the flag has to be set to the value 1.

4. The programmable FIFO almost full flag (PAFF): The PAFF can be set by the user to any value between 32k and 64k. The value is programmed by giving a number between 0 and 32k; the programmed value is counted with respect to the upper FIFO edge. E.g. for a value of 1000, the PAFF is programmed to the value 64k-1000. The PAFF becomes active if there are equal or more words than the programed value written into the FIFO. For telescope operation, the PAFF is used as EBF (see the section about IRQC for details).

There is another flag, the FIFO half full flag, made available by the FIFO, but this flag is not connected to the MFPGA and not used in telescope operation. All other control signals are tied to fixed voltage levels, so the user doesn't have to worry about them.

[76] This may change for a later version of the DTC card.

Resetting the FIFO When a FIFO reset is given, all information contained in the FIFO is erased and also the programmed flag values[77]. So, care has to be taken to re-program both, PAEF and PAFF to the old values (otherwise one might get into serious trouble). A reset is given to the FIFO by giving a MSQ reset (there is no separate reset for the FIFO only). An MSQ reset is given by setting the *SEQUENCER-MASK* in CMD I (access to CMD I with MA 9) to the value 0. Please note that, after the reset is finished (this doesn't take long) the *SEQUENCER-MASK* has to be reset to either 2 (MSQ is inactive) or 1 (MSQ is active). Please note that at the moment a reset is given to the FIFO, DPR and MSQ are as well reset and have to be re-activated then before normal operation can resume.

Programming PAFF and PAEF To program PAFF and PAEF, two 16 bit words have to be written into the FIFO with the FIFO LOAD signal active. The FIFO LOAD signal is set by setting CMD I bit 4 active. The 2 values which are to be programmed, have to be prepared in the FIFOHI and FIFOLO registers in the FICO each. This is done in the same way as writing into the FIFO for test purposes, so the TESTMODE (CMD I, bit 7) has to be active. The first write access with the FIFO LOAD active after a reset will write into the PAEF register, thus programming the PAEF value. The second write access to the FIFO (with FIFO LOAD active) will then write into the PAFF register and program the PAFF value. After programming both values, the FIFO LOAD has to be de-activated again to make the changes become active.

7.5 The module's RAM

In this section the module's RAM and its features are described, as far as they are important for the telescope operation. For more information, please consult the RAM datasheet. The RAM is a fast CMOS static RAM, Cypress CY7C1021, IDT 71016 (or equivalent type) in a 44 pin standard plastic SOJ package. The RAM is 16 bit wide and 64k deep. In normal telescope operation it is used to make threshold and pedestal values prepared during DAQ setup available for the DPR (not the full RAM is used, only one word for each FE-channel)[78]. In this operation mode, the control of the RAM is transferred to the MSQ. To prepare the RAM for data acquisition, it has

[77] They are reset to built-in default values; 127 for PAEF and 64k-127 for PAFF.

[78] For this purpose a smaller RAM would have been sufficient, but in previous versions of the telescope setup the RAM was used to generate the readout sequences.

to be written byte-wide. For test purposes, it can also be read byte-wide. Byte-wide access is necessary, because with each read or write access the RAM address counter implemented in the MSQ is incremented, but only an 8 bit data word can be transferred to or from the PC. While operating the RAM in this mode, access is controlled by using the MA 8 and MA 9 (RAM read hi and RAM read lo) or MA 10 and MA 11 (RAM write hi and RAM write lo) respectively.
Whenever MA 8 or 9 is active and a write access occurs, the data written is directly transferred to the RAM byte selected by the current value of the RAM address counter and the mode address (deciding whether hi or lo byte is to be written). For the read access, it is exactly the same. Outside the MFPGA, RAM access is controlled by the RAM output enable (corresponding to RAM write hi or RAM write lo), the RAM write enable (which is basically generated by the WE-B used inside the MFPGA) and RAM write hi and RAM write lo themselves. The RAM output is always 16 bit wide, but can only be accessed via Tri-state Buffers connected to the 8 bit data bus inside the MFPGA in the way described above. Please note that for extremely long cable lengths there might be a problem with the RAM test as the timing of the RD-CLK which is used to increment the RAM address counter during a read access is not optimized for long cable lengths. This problem is known and understood, an in a later version of the MFPGA configuration it will be fixed. And, as only the test readback is concerned[79], the problem can be ignored confidently.

[79]This has been proven by checking the correct functionality of RAM dependent DPR tasks.

8 The ATC

A view of the ATC showing the component locations is shown in fig. 38. A photo of an ATC inside a module setup is shown in fig. 39. The ATC is the analog pendant to the DTC. As the DTC holds all digital controls, generates all readout sequences and supervises the communication to the DAQ PC, the ATC controls all the analog stuff necessary to successfully operate a telescope module. One side of the ATC is connected to the DTC via the DTC port, the other side is connected to the two VA-hybrids located on the module's hybrid assembly (see section about front-End electronics). The ATC is a 4 layer PCB which contains the following functional building blocks:

- Hybrid voltage supplies (SVH)
- Hybrid bias generation (HB)
- Detector bias voltage handling (HV)
- Signal level shifters (LS)
- Current-to-voltage converters (I2UC)
- Fast digitization of analog output data (DGS)
- Data output multiplexers (DOM)
- Calibration chopper (CAC)
- Monitoring circuits (MOC)
- Clock control (CLC)

As the ATC has to control 2 hybrids which are directly connected to the detector, it has to provide the building blocks listed before twice, once for every detector side. As the two detector sides are named N- and P- side, depending on doping and detector contact type, the ATC part controlling the hybrid connected to the N-side is called N-compartment, the part controlling the hybrid connected to the P-side is called P-compartment. As N- and P-compartment do not differ in principle, the circuit features are only described once. Pin- and signal assignments are the same unless otherwise noted. The simplified flow chart of one ATC compartment is shown in fig. 40.

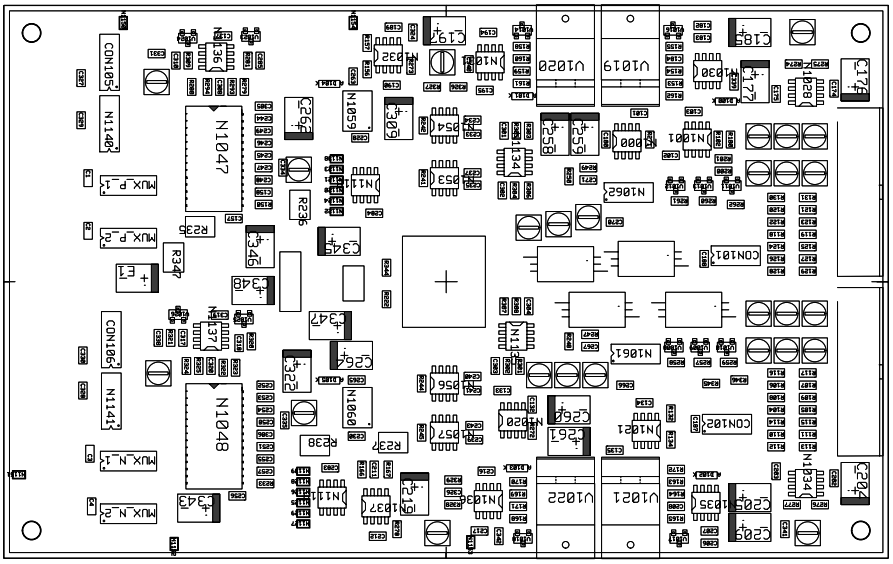


Figure 38: ATC view showing component locations. The DTC port is on the left side, the connectors to the front end hybrids can be seen on the right side. The big white capacitor in the middle is the HV bias decoupling low-pass capacitor. The axial symmetry of the circuit due to two identical circuits being on the same PCB can be seen in the picture

Figure 39: Photo of an ATC view inside a module setup. The orientation is the same as in fig. 38.

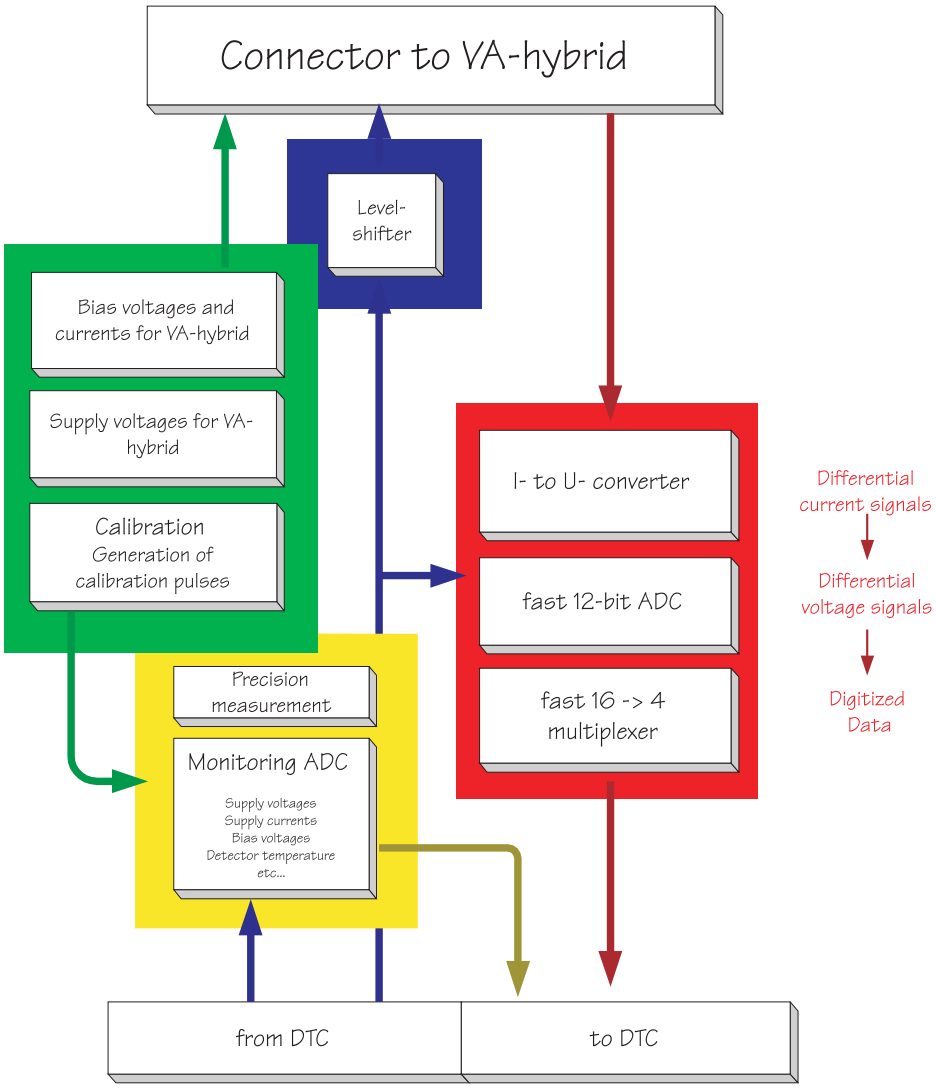


Figure 40: Simplified circuit flow chart of one ATC compartment. The different functional building blocks are color coded. *Green:* Bias and supply. *Yellow:* Monitoring. *Blue:* Digital stuff. *Red:* Analog signal processing.

8.1 PCB features

The details of the circuit can be taken from the schematic and layout printouts from the documentation folder. Although 4 PCB layers were available, routing of analog and digital signals is done mostly on the solder and component side of the PCB. The inner layers are used for transferring the analog and digital supply voltages and grounds, also providing an efficient shielding. The DTC-ATC connection is made via a 50 pin low profile connector. Inside

the module's case, the ATC is sitting right on the back side of the DTC, both PCBs forming kind of a PCB sandwich. The connection between ATC and hybrid is made via 34 pol. high density connectors for IDC cables, one for each compartment. The power connection is made by low profile PCB interconnect connectors, one 6 pol. connector delivering $2 \times \pm 5$ V supply voltages, one pair of voltages for each ATC compartment, and one 2 pol. connector delivering the detector bias HV to the special circuit section dealing with it and delivering it to the detector.
As the circuit handles both, low noise analog signals and high speed digital signals with large voltage swing (TTL levels), the design has been optimized for low analog-digital cross talk to prevent additional contributions to the system's noise. Therefore, the very sensitive circuit part responsible for current-to-voltage conversion[80] (I2U converter) and digitization of the analog voltage signals is spatially separated from any signal lines carrying high speed digital signals (like clocks etc.) and specially shielded. The lines carrying the differential current signal are not lines on the PCB itself; the current signal is transmitted from the hybrid connector to the I2Uconverter via a specially shielded 2-wire differential cable fixed[81] to the back side of the ATC directly above a large ground plane. The way how to do this is displayed in fig. 41.

Although the detector is AC-coupled to the VA 2 front end ICs it has turned out that operation the front end electronics with its ground level on detector bias voltage yields a much better performance, especially concerning the behaviour for pin-hole channels[82] etc.. Therefore, the whole ATC compartment connected to the hybrid is on *bias ground*. This results in the condition that, as each compartment is on its detector side's bias ground, the two compartments, although spatially separated by a gap of few millimeters, are separated by the detector bias voltage. **Care has to be taken here not to accidentally short the HV bias**. When debugging the analog circuits, please follow these instructions:

- Don't panic!
- Turn off the HV bias, if possible. Only debug on an ATC compartment while HV bias is on if it is absolutely necessary.
- After turning off the HV bias, wait until the voltage dropped to a low level before proceeding.

[80] The VA 2 ICs analog output signals consist of a differential current signal.

[81] We fixed the cable by using openings in the solder resist on the back side of the PCB to solder small bridges of thick wire and threaded the cable through the openings.

[82] These are channels in which the strip insulating capacitor is shortened.

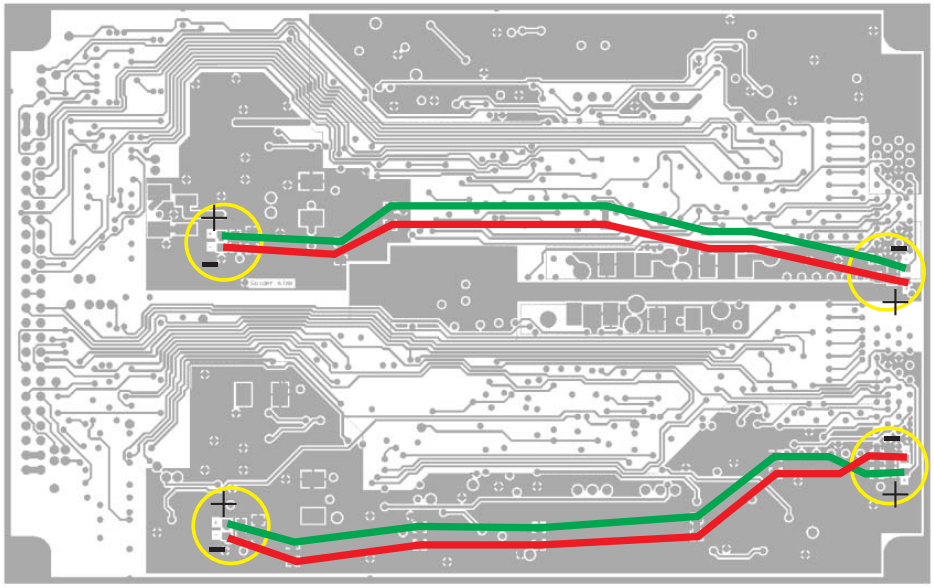


Figure 41: Adding the ATC analog signal lines to the ATC backplane. Please note the DPR needs positive signal polarity for both detector sides. As I2UC and ADC (the latter due to a twist in the connections) both invert the signal, the cables for the N-side have to be twisted.

- If you want to compare behaviour of the two compartments e.g. by measuring the properties of the two compartments using the same device, make sure the ATC compartment grounds are connected. Otherwise you will only measure rubbish. Create an artificial short if necessary (e.g. by bridging the HV bias connector).
- If it is not possible to avoid measuring on a biased ATC, be careful not to accidentally shorten the bias voltage.
- Never measure properties of two compartments of a biased ATC simultaneously without using appropriate equipment (HV probe etc.). Not doing this may result not only in damage to the ATC and the measurement equipment, you may also be injured.
- When measuring the properties of a biased ATC with equipment directly connected to the power line: Be careful not to accidentally shorten the bias HV to "world ground". Please check which compartment has a connection to world ground before proceeding (see below). Use a power line separation transformer if necessary.

As mentioned before, the ATC compartments are connected their bias ground. They are galvanically insulated from each other and, what is more, via the

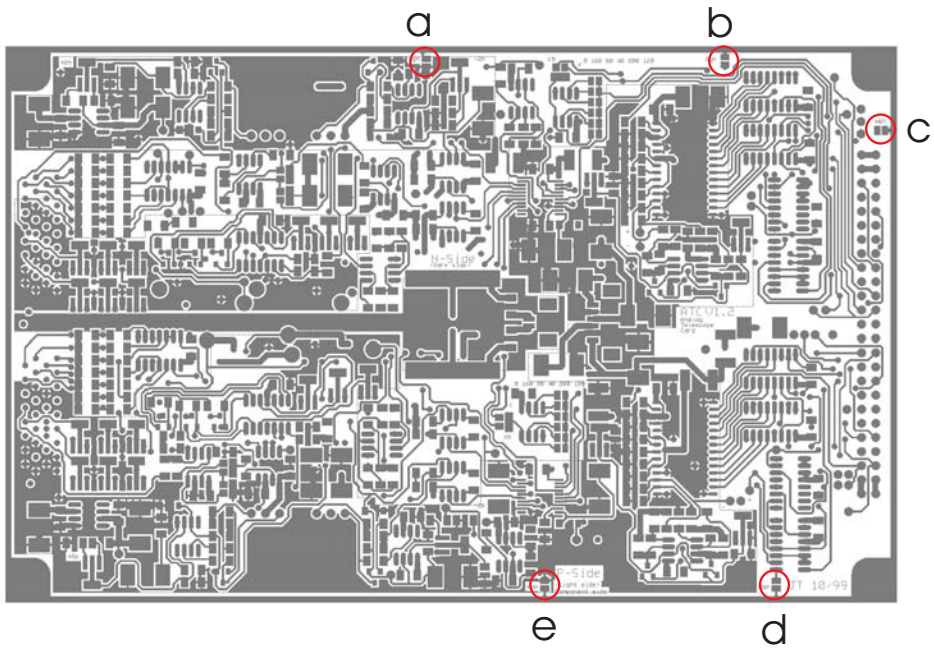


Figure 42: Location and assignment of ATC grounding pins. *a*: connection to analog ground P, *b*: connection to digital ground P, *d*: connection to digital ground N, *e*: connection to analog ground N. The jumper *c*: connects the frame around the ATC with world ground (DTC digital ground), leaving the ATC compartments floating. Never connect jumpers of N- and P- side with the frame at the same time (detector bias voltage will be shortened).

coupler's bench from the DTC world ground. However, it has turned out that for best common mode noise behaviour of the system it is the best if one compartment has one of its ground voltages somehow connected to world ground. Each ATC compartment has two different ground connection, analog ground and digital ground. For each ground a jumper has been provided, with which a connection to world ground can be made. The position and assignment of these jumpers is shown in fig. 42. The best results have been achieved by connecting the analog ground of the N-compartment to world ground using a 100 kΩ series resistor.

As described, the HV is connected to the ATC using a low profile, 2 pin connector right in the middle of the ATC card. Although the ATC grounds are on bias level, both polarities of the HV are not directly connected to the corresponding compartment ground on the ATC. In fact, this connection is made on the hybrid assembly (see the description of the front end electronics). Thus, as long as there is no hybrid assembly connected, the ATC compartments are not connected to bias ground; they are floating. On the ATC, the HV is low-pass filtered and then transferred to the connector

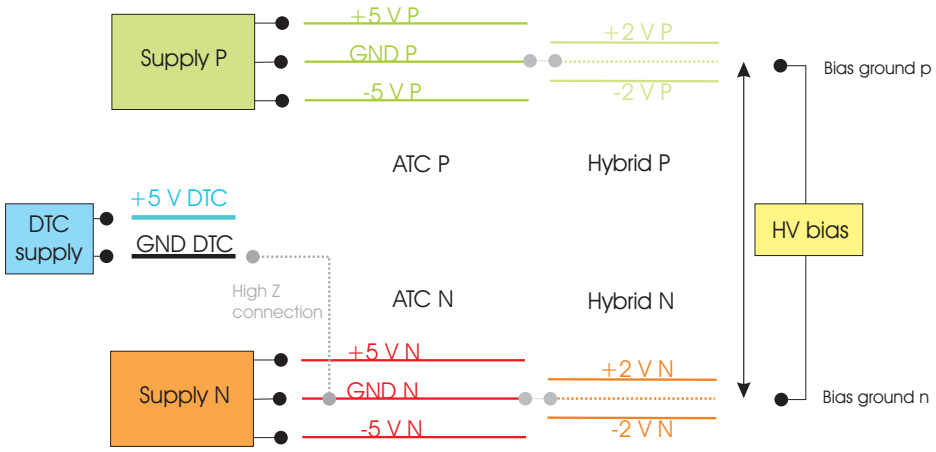


Figure 43: Potential levels in a telescope module. The connection between DTC ground and GND N is one of four possibilities and the one we found the system to operate with the lowest common mode noise with. The connection between the ATC grounds and bias ground is only present if there is a hybrid assembly connected to the module.

to the hybrid assembly via two broad power lines buried in the inner layers of the ATC PCB between two shielding layers which can be connected to any shielding potential wanted using small openings in the solder resist right in the middle of the PCB. Again, the best results have been achieved by connecting the shielding layers to the N-compartment analog ground, this time using solid metal contacts for the connection. Fig. 43 gives an overview about the connections between the different potential levels and their connections during normal telescope operation.

Due to software problems with the CAD system that was used for designing the PCB, there are missing connections at two components on the N-compartment. These had to be wire-wrapped afterwards. Also, some modifications had to be made at the I2U-conversion stage, which also required some bricolage. Details of this operation can be taken from fig. 69.
In the following sections, the operation principles of the ATC's most important circuit sections are described without going too much into detail. They are meant as a help to understand the circuit schematics. For each building block the page of the circuit schematics is given. For details about the arrangement of components on the PCB please refer to the circuit layouts.

8.2 Hybrid voltage supplies (SVH)

ATC schematics page 5 and 6. The VA2 hybrids need a supply voltage of ± 2 V. The ATC compartment receives supply voltages of ± 5 V form the module's power supply. The hybrid voltage supply generates the hybrid voltage supply from these ± 5 V, allows adjustment of the voltage in some range around ± 2 V and has implemented features like over-voltage protection and current limitation. It is also possible to perform online monitoring of voltage and current draw for both supply voltages (together with the MOCO section). A standard, low-drift voltage reference is used to generate the 2.5 V reference voltage for the SVH. The negative reference voltage is generated by simply inverting this voltage with an inverting amplifier. For both the negative and positive supply voltage the reference voltage is fed trough a voltage divider with a potentiometer, whose output serves as reference (non-inverting) input of an operational amplifier which drives the base of a power npn (power pnp respectively, for the positive supply) transistor, whose collector is tied to ± 5 V. The emitter is connected via 2 series resistors (one of which is a high precision resistor for current measurements) whose values define the maximum current for the current limitation circuit to the ± 2 V power lines. If the voltage drop over those resistors becomes too high due to a large current flow on the corresponding supply voltage, further current draw will be limited. A 2.6 V Zener diode connected to ground provides the over-voltage protection. The components involved in this circuit section are analog ground and analog ± 5 V, and a lot of decoupling capacitors are used to provide low noise voltages.

8.3 Hybrid bias generation (HB)

ATC schematics page 9. As can be seen in the section about front end electronics, each VA needs 3 bias currents and 2 bias voltages. On the VA hybrids, 5 VA 2 ICs are together on a hybrid. All three bias currents and one of the bias voltages are provided once for all VAs on a hybrid. The remaining bias voltage can be adjusted individually. So on an ATC compartment 3 bias currents in quite different ranges have to be supplied, one bias voltage for all VAs in common and one bias voltage for each VA-IC. All six bias voltages are generated by simply taking the middle connector of a SMD precision potentiometer. The bias currents are generated by three voltage programmable current sources, each having a current to voltage ratio adjusted to the bias current range. The controlling device is a quad, low noise operational amplifier. As the current sources must source (not sink) current, pnp transistors

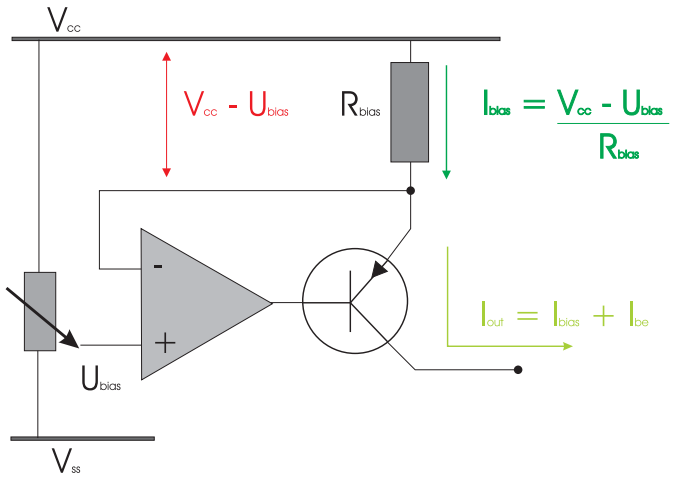


Figure 44: Schematic of a bias current source. The OpAmp is one of four OpAmp circuits made available by a Maxim quad OpAmp package. The bias current is defined by U_{bias}, which can be adjusted by an external potentiometer. Depending on the value of R_{bias} the OpAmp regulates the conductivity of the transistor so, that the emitter of the transistor lies at U_{bias} voltage level, inducing a current through the bias resistor only defined by the voltage difference between V_{cc} and U_{bias}. This is exactly the current which can be taken from the current output (the base-emitter current is about two orders of magnitude smaller).

are used as regulators. In fig. 44 a schematic of a bias current generation circuit is given. The bias current flowing can be calculated from the following formula:

$$I_{bias} = \left(\frac{V_{cc} - U_{bias}}{R_{bias}}\right)$$

Table 17 gives an overview about the bias current source demands and properties. The programming voltage for the current sources can be adjusted by three SMD precision potentiometers located close to the OpAmp circuit.

8.4 Detector bias voltage handling (HV)

As described above, the detector bias voltage is delivered to the ATC card via a low profile SMD connector. To remove pickup from the voltage, the HV is low pass filtered right afterwards. Pickup on the HV will increase common mode noise, so it is important to suppress it. The filtered voltage is passed to the hybrid connectors via two lines on the inner layers in the middle of the card, right between the compartments, shielded by two shield layers, on top and on the bottom, which can be set to an arbitrary potential. It turned out

Parameter	Description	R_{bias}	U_{bias}	Value
I_{buf}	Output buffer bias current	5.6 kΩ	0.88 V	200 μA
I_{sha}	CRRC shaper bias current	5.6 kΩ	1.384 V	110 μA
I_{pre}	Adjusting operating point for FE preamplifiers	300 Ω	1.25 V	2.5 mA

Table 17: Overview over the different bias current sources. The bias currents are global for the hybrids. Except for I_{buf}, which always flows only once, all values are five times the nominal values. The current values are given for $V_{cc} = 2$V.

that the best results could be obtained by connecting both HV shield layers to the analog ground of the N-compartment. The HV bias has to be low-pass filtered once again, on the hybrid itself, at the point where the connection to the N- or P- compartment ground is made. More on this in the section about the hybrid assembly. The locations of all potentiometers to adjust bias parameters is shown in fig. 49 at the end of this section.

8.5 Signal level shifters (LS)

ATC schematics page 3. Several digital control signals are transmitted from the DTC to the ATC. Some of them are used on the ATC only, some others are to be transferred directly to the front end hybrids. But as the digital signals are TTL and the front end hybrids expect digital signals with a swing around ground from - 1 V to + 1 V, the digital signals have to be level-shifted. There are 5 digital signals which go to the hybrids. 2 of them, the *hold* and the *clock* are pseudo-differential, that means that there are 2 signal lines for this signal, one for the signal itself and one for the inverted signal. The input stage on the VA hybrid is not differential though, but using signal and inverted signal at a time minimizes signal feed-trough to sensible analog output signals, and this is important here because *hold* and *clock* are the only signals which are to be clocked with the front end amplifiers being sensitive. The inverted signal is generated by an inverter / driver IC which can provide up to 100 mA output current. To make signal and inverted signal be driven equally, the inverted signal is fed into another channel of the IC to generate the original signal polarity. The TTL output signals now have to be level shifted. The level shift itself is done with a voltage divider using 2 1 kΩ resistors in row, whose lower end is connected to the -2 V supply and the

signal output connection being in the middle, providing a voltage level of ≈ +1 V if the signal is active and a level of ≈ -1 V if the signal is inactive. The inverter/driver IC has 6 channels, 4 of which are used for the *hold* ad *clock* signals. From the remaining signals, 2, *data in* and *test on*, are fed trough the inverters (thus being inverted once[83]) and then level-shifted the same way as *hold* and *clock* are. The remaining signal, *digital reset*, has no connection to a driver channel, but is level shifted the same way. In an improved version of the ATC, one would either connect all digital signals with drivers or at least connect the *digital reset*, which is a signal that has to be switched during each readout cycle, via a driver channel. *Test on* is only needed in calibration mode and when preamplifier parameters have to be adjusted, and there is no need to switch this signal fast.

8.6 Current-to-voltage converters (I2UC)

As is said in the section about the front end electronics, the analog output signal of the VA hybrid is a series of differential current signals. The fast digitization ADC used needs differential voltage signals as inputs. So the differential current signal has to be converted into a differential voltage signal. This conversion has 2 constraints:

1. The VA output signal lines should be kept at a potential of -1 V. Measurements have shown that the output signal swing is largest if this constraint is met.

2. The analog output signal of the I2UC has to be adapted to the input voltage range of the two signal inputs of the ADC.

The first constraint is met by simply using an operational amplifier as I2UC for each polarity of the VA output signal[84]. The VA output signal is connected without any series resistors directly with the inverting input of the OpAmp, the non-inverting input is connected with a fixed reference potential U_{ref}, which, of course, for this application is set to -1 V. The golden rule for

[83] One has to be careful not to mess up the signal polarities!

[84] Originally a cascode was implemented on the version of the ATC card currently in use to keep the potential at the VA output even more stable at -1 V. But this concept has been abandoned later due to problems with the dynamic range. Some changes had to be made to the ATC card to change back to the simple OpAmp I2UCs. These changes are documented in the appendix.

the operational amplifier then yields

$$-\frac{U_{out} - U_{ref}}{R_{feedback}} \;=\; I_{in}$$

Thus, the output voltage is only defined by the feedback resistor providing the desired gain and an constant offset voltage:

$$U_{out} \;=\; U_{ref} \;-\; R_{feedback} \;\cdot\; I_{in} \tag{3}$$

With $U_{ref} \;= -1V$ the feedback will take care of keeping the inverting input and thus the VA buffer output at the desired voltage. Obviously, as the input current is connected to the inverting input, the output is inverted. A larger signal yields a more negative output voltage.
The next constraint is a bit harder to meet. An overview about the circuit schematics is given in fig. 45. The fast ADC accepts input signals in a range between $U_{ADC}^{Min} = +1.25V$ and $U_{ADC}^{Max} = +3.25V$. Each polarity of the differential output signal of the I2UC has to be moved into this range. This can be done by adding an "offset current" to the signal current for each side independently. Adding a current to the signal is easy here, because the voltage on the signal line is fixed by the OpAmp. Thus, a current is added by connecting a variable resistor to the signal lines actively kept at U_{ref} on one side and to a certain constant bias potential on the other side, whose value depends on sign and magnitude of the offset current needed. The locations of the potentiometers determining the offsets is shown in fig. 49. The offset current will be added to the signal output current:

$$U_{output} \;=\; U_{signal} \;+\; U_{output}^{offset} = U_{ref} - R_{feedback} \;\cdot\; \left(I_{signal} \;+\; I_{output}^{offset}\right)$$

By regulating the offset current, the output voltage of the I2UC can be adapted to the input voltage range of the fast ADC.
As explained in the section about front end electronics, the signals generated by the VA2 ICs consist of a default buffer current, the buffer bias I_{buf}, which has to be provided from the outside. This current flows if the serializer of the VA hybrid is initialized. This current splits into two parts of equal size each flowing through one line of the differential output buffer. Thus, the default current I_{idle} observed without any signal is:

$$I_{+}^{idle} \;=\; \frac{I_{buf}}{2}$$

$$I_{-}^{idle} \;=\; \frac{I_{buf}}{2}$$

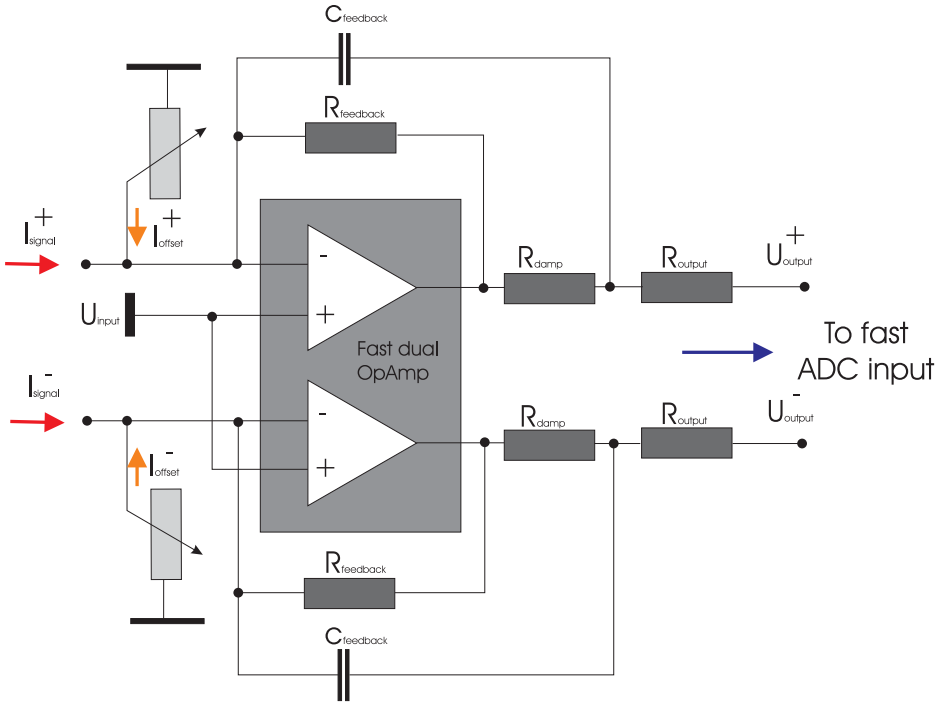


Figure 45: Circuit schematics for the I2U converter circuit. The current inputs directly come from the connector to the FE hybrid, the output voltages are directly connected to the fast ADC input. Except for the different offset current settings the circuits for both branches are absolutely identical. R_{damp} and $C_{feedback}$ prevent the I2Uc from oscillating, R_{output} is just a series resistor for the ADC input. The values are: $C_{feedback}$ =3.9 pF, $R_{feedback}$ =7.5 kΩ, R_{damp} = 10 Ω, R_{output} =10 Ω. The offset current potentiometers are multi turn trim potentiometers with an absolute resistance of 20 kΩ.

In case there is an additional signal current[85], the signal current modulates the default current in the following way:

$$I_+ = I_+^{idle} + \frac{I_{signal}}{2} = \frac{I_{buf}}{2} + \frac{I_{signal}}{2}$$

$$I_- = I_-^{idle} - \frac{I_{signal}}{2} = \frac{I_{buf}}{2} - \frac{I_{signal}}{2}$$

Thus the total signal can be recovered by

$$I_{signal} = I_+ - I_-$$

The I_{buf} defines the linear range of the VA output. This current is always positive, flowing out of the VA and into the input of the I2UC circuit as

[85] As will be explained later, this is always the case, because no channel has "no signal".

displayed in fig. 46. The signal current can't be larger than that buffer bias current. Thus, the output current of the VA's analog output buffer is always positive in either branch independent from the signal current.
The signal current consists of two contributions. First, there is the *pedestal signal*, which contains the information about the pedestal value, that is the default analog output signal with no charge present at the FE input for the considered channel. Additionally, there is the *event signal*, which actually contains the information about signal charge collected for an event at the FE input in this channel.

$$I_{signal} = I_{event} + I_{pedestal}$$

The pedestal signal $I_{pedestal}$ can be of either polarity. The pedestal values for different channels across a FE IC and across a whole hybrid will vary a lot; also, one channel's pedestal value will fluctuate[86]. The event signal's polarity depends on the sign of the charge carriers collected, which is defined by the detector side the VAs are connected to. For a VA connected to a detector's N side (positive bias voltage) electrons will be collected and the event signal is negative. Signals from the P side of the detector generate a positive event signal current.
The signal currents are fed into the I2UC. As the noninverting input is kept at $U_{ref} = -1V$, and the a constant offset current is added to either side in the following way:

$$I_{+}^{offset} = \frac{U_{Ref} - U_{OB}}{R_{+}^{offset}}$$

$$I_{-}^{offset} = \frac{U_{Ref} - U_{OB}}{R_{-}^{offset}}$$

The voltage difference is fixed, so the current is defined only by the value of the offset resistor. As the offset bias voltage U_{OB} is chosen to be -2 V (V_{ss}) the offset current will be negative. This is necessary, as the VA output current being always positive forces the I2UC output to become even more negative according to equation 3. What is finally fed into the I2UCs is the following:

$$I_{+} = \frac{I_{buf}}{2} + \frac{I_{pedestal} + I_{event}}{2} + I_{+}^{offset}$$

$$I_{-} = \frac{I_{buf}}{2} - \frac{I_{pedestal} + I_{event}}{2} + I_{-}^{offset}$$

[86] The σ of one channel's pedestal fluctuations is what we call this channel's noise, and, of course, common mode fluctuations in either direction also have to be taken into account here.

According to equation 3, the output voltage levels of the I2UC are defined by feedback resistors and reference voltage:

$$U_+ = U_{ref} - R_{feedback} \cdot \left(\frac{I_{buf}}{2} + \frac{I_{pedestal} + I_{event}}{2} + I_+^{offset} \right)$$

$$U_- = U_{ref} - R_{feedback} \cdot \left(\frac{I_{buf}}{2} - \frac{I_{pedestal} + I_{event}}{2} + I_-^{offset} \right)$$

Combining the constant terms, one gets:

$$U_+ = U_+^{offset} - \frac{U_{event}}{2} - \frac{U_{pedestal}}{2} \quad (4)$$

$$U_- = U_-^{offset} + \frac{U_{event}}{2} + \frac{U_{pedestal}}{2} \quad (5)$$

Using the following definitions

$$U_+^{offset} = U_{ref} - R_{feedback} \cdot \left(\frac{I_{buf}}{2} + I_+^{offset} \right)$$

$$U_-^{offset} = U_{ref} - R_{feedback} \cdot \left(\frac{I_{buf}}{2} + I_-^{offset} \right)$$

$$U_{event} = R_{feedback} \cdot I_{event}$$

$$U_{pedestal} = R_{feedback} \cdot I_{pedestal}$$

These definitons are chosen in a way which preserves the original sign of the signal current. Considering only VA ICs connected to one detector side, one has only event signal currents with a defined sign. Neglecting the contributions from the pedestal signals, it is a good idea to adjust I_+^{offset} and I_-^{offset} so, that, depending on the signal polarity, the default values for U_+ and U_- is at the upper or lower edge of the fast ADC input voltage range, respectively. For example, when dealing with signals from the N detector side, the signal current will be negative, and from equations given in 4 and 5 it can be seen that U_+ will increase and U_- will decrease if there is some signal in a certain channel. So the best values to adjust the offsets are

$$U_+^{offset} = U_{ADC}^{Min}$$

as the I2UC output voltage will increase, and

$$U_-^{offset} = U_{ADC}^{Max}$$

as the output voltage will decrease (when considering signals from the P side of a given detector, it is just the other way round). $I_{pedestal}$ is not determined

by the detector side the channel is connected to but mainly by the channel's properties depending on the FE IC settings. So the offset voltages have to be adjusted in a way that leaves some space for pedestal variations and fluctuations in "the different direction", the reverse polarity of the expected signals.

$$\begin{aligned} U_{+}^{offset} &= U_{ADC}^{Min} + U_{+}^{Margin} \\ U_{-}^{offset} &= U_{ADC}^{Max} - U_{-}^{Margin} \end{aligned}$$

The margin voltages have to be adjusted with respect to the module's pedestal fluctuations and variations. Also, the dynamic behaviour of the I2UC has to be regarded (oscillations, spikes etc...). If the margins are too large, more dynamic range than necessary will be lost, if the margin is too small, signal amplitude or even entire events might get lost.
The current to voltage gain of the I2UC is solely defined by the value of its feedback resistor. It has to be chosen with respect to the values of I_{buf} and I_{event} and the input voltage range of the fast ADC. As pointed out in the section about FE electronics, the charge to current gain of the preamplifiers is ≈ 25 μA / fC. An average event deposits an amount of ≈ 4 fC of charge in the detector, so the event signal current on the detector N side is -100μA, which yields a voltage signal of 760 mV. A typical value for the buffer bias current is 200 μA (the nominal value is a bit smaller, but this is the value the VAs are operated with in the BAT module setup). Offset current and I2UC feedback resistor have to be chosen so that

$$U_{ref} - R_{feedback} \cdot \left(I_{+}^{offset} + \frac{I_{buf}}{2} \right) - \frac{U_{pedestal}}{2} \geq U_{ADC}^{Min} \qquad \text{for } U_{+}$$

$$U_{ref} - R_{feedback} \cdot \left(I_{-}^{offset} + \frac{I_{buf}}{2} \right) + \frac{U_{pedestal}}{2} \leq U_{ADC}^{Max} \qquad \text{for } U_{-}$$

Assuming an absolute pedestal voltage of 10% ($\approx$ 80 mV) of the signal voltage in opposite signal direction, covering pedestal fluctuations, common mode and everything, one can calculate an estimated set point for I_{+}^{offset} and I_{-}^{offset}:

$$\begin{aligned} I_{+}^{offset} &= \frac{\left(U_{ref} - \frac{U_{pedestal}}{2} - U_{ADC}^{Min} \right)}{R_{feedback}} - \frac{I_{buf}}{2} \\ I_{-}^{offset} &= \frac{\left(U_{ref} + \frac{U_{pedestal}}{2} - U_{ADC}^{Max} \right)}{R_{feedback}} - \frac{I_{buf}}{2} \end{aligned}$$

The calculation yields ≈ -350 μA for I_{+}^{offset} and ≈ -610 μA for I_{-}^{offset}, corresponding to offset resistor values of $R_{+}^{offset} = 2.8$ $k\Omega$ and $R_{-}^{offset} = 1.6$ $k\Omega$. It has to be stressed here, that the given values and current signs

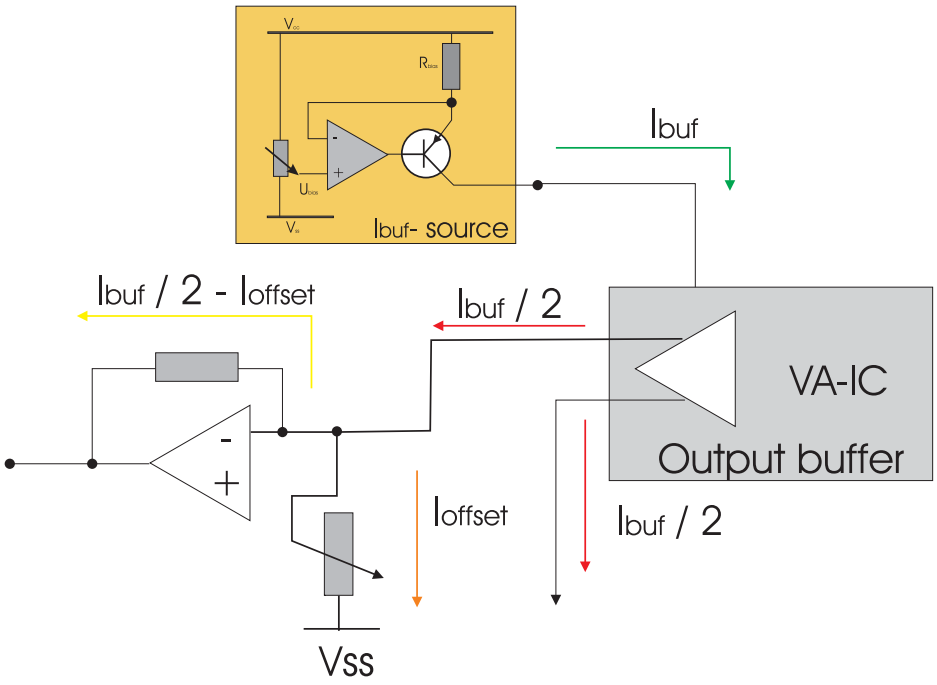


Figure 46: Flow of the VA buffer bias current. The current, modulated with the signal amplitude, always flows out of the IC and into the I2UC circuit input. Adding a constant offset current to the signal current allows an adaption of I2UC output to the fast ADCs input voltage range.

are given for N side signals, but one can easily repeat those calculations for P side signals (it is just the other way round). By the way: the offset voltages do not modify the event amplitude measured by the data preprocessing units. As the pedestal current for each channel and the total offset current difference should be nearly constant, the offset current acts as an artificial additional pedestal value. This constant contribution is always subtracted during data preprocessing, as long as the voltage difference which defines the offset current is not modified. Otherwise, one might get an additional small contribution to the system's noise.

8.7 Fast digitization of analog output data (DGS).

The DGS stage of the ATC consists of the fast ADC and a clock controlling stage. The fast ADC used is a Burr-Brown ADS 802 U pipeline ADC. The clock controlling stage controls the timing of the ADC conversion clock and consists of a digital delay circuit which can delay an incoming digital pulse

by 200 ns in 5 steps with 20 ns delay each. As the maximum clock frequency the system is operated with is 5 MHz corresponding to a period of 200 ns, the phase between shift and sampling clock can be adjusted with sufficient precision using this delay circuit. The actual delay can be selected using a 5 pin jumper system. Moreover, one can choose whether the clock itself or the inverted clock will be used. This clock controlling section is necessary to set up the right timing between the incoming analog step-shaped signal and the conversion clock, as it is illustrated in fig. 47. The locations and assignments of the jumpers are shown in fig. 48. The ADC is a 12 bit pipeline ADC. This means that, as the digitization inside the ADC takes place in several steps the data for sampling clock cycle N, which is sampled with the *falling* edge of clock pulse N appears at the output 6 $\frac{1}{2}$ clock cycles later[87], that is, with the rising edge of clock cycle $N+6$. The signal processing circuits following after the ADC output have to take this into account. The small delay necessary to adapt the the ADC sampling clock to the VA output signal also plays a role. A change in the delay will entail major changes in the sequencer programming of subsequent signal processing logic. So, making changes to the phase shift is a major act and care has to be taken not to unintentionally change the DGS delay.

8.8 Data output multiplexer (DOM).

As explained in the section about the DTC PCB, there is only a limited number of channels available or the communication between an ATC compartment and the DTC. As far as the data transfer in the direction from ATC to DTC is concerned, 4 channels are provided for each ATC compartment. Thus, the 12 ADC bits have to be multiplexed. Because there is more data than only the ADC data one might want to transfer to the DTC, a 16 to 4 multiplexer has been implemented on the ATC card. 16 bit in total can be transferred to the DTA, 12 of which are, of course, the ADC output data bits. The other four ones are:

- *DTO: Data out.* This is the serial data output of the 8 channel monitoring ADC, which is described in the section about monitoring. This bit is not needed for fast data acquisition, the monitoring ADC is only operated in software sequencer mode.
- *DOD: Data out DAC.* The DAC used to generate the VA hybrid calibration voltage is also operated serially. Its serial shift register can

[87] This is the so-called data latency.

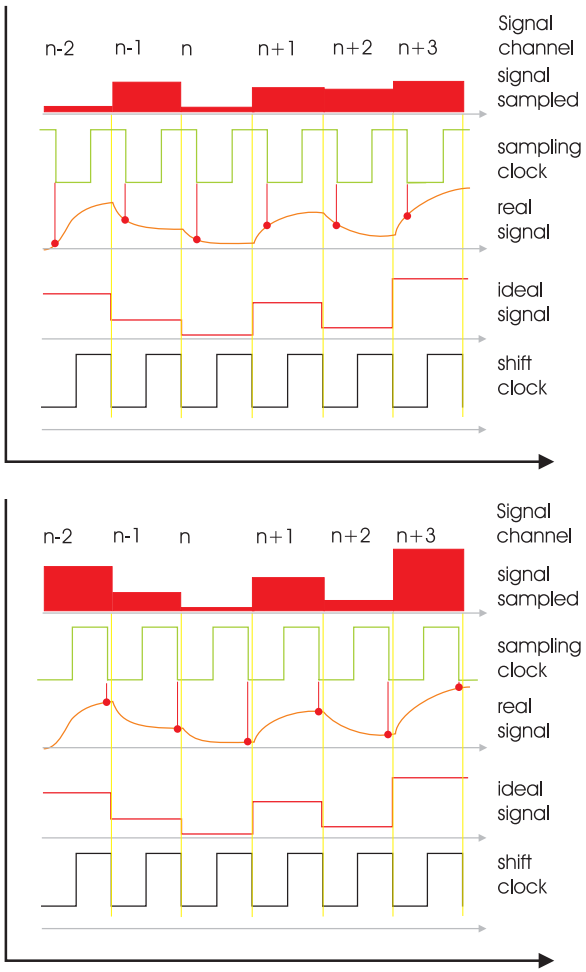


Figure 47: Timing of telescope shift and sampling clock on ATC. The clocks are active low. The signal shape is not rectangular as would be expected for an ideal system. Parasitic capacitances and finite rise times of buffers etc. act as an effective low pass. This effect can partially be compensated by carefully adjusting the timing between shift clock (which causes a new signal step to appear at the ADC input) and the sampling clock, which transfers the new signal into the ADCs internal signal processing circuit. Upper part: Bad timing. The signal has no time to develop. In an extreme case, these conditions lead to misallocation of hits and signal loss. Lower part: Ideal timing. Signal loss is minimized as the signal has enough time to develop.The timing can be adjusted by setting the clock control jumpers of the DGS circuit part. Remaining RC rest effects in the signal can be corrected by applying a RC deconvolution algorithm on the data.

be read back using this bit to check communication between ATC and DTC and to test the DAC. As the DAC is also operated in SoSe mode

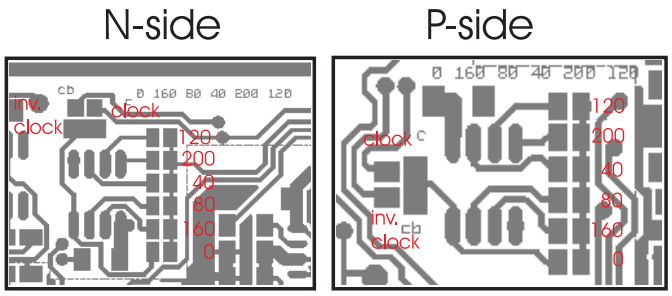


Figure 48: Jumpers for clock control and timing adjustment of shift and sample clock of the DGS circuit. The numbers connected with the jumper pads correspond to the selected delay value / clock phase. delay values are given in ns. The labels on the PCB can also be used to identify the meaning of the applied jumper settings once one is familiar with the order. It turned out, that the best delay value for the current ATC and VA hybrid configuration is *clock* and *200 ns*.

only, this bit is discarded during fast data acquisition.

- *SST: Serial strobe.* The serial strobe indicates that the monitoring ADC has accepted a command sequence and finished processing of the analog input data. The serial strobe goes from Hi to Lo when the conversion starts, and goes back from Lo to Hi again, if the conversion has been finished. Thus, when operating the monitoring ADC the serial strobe indicates when there is data to be read back from the monitoring ADC and if the monitoring ADC at all accepted the command word sent. Like the other two DAC and ADC bits, this bit is discarded during fast data acquisition, too.

- *SHO: Shift out.* The 5 VAs on a front end hybrid are connected together forming a *daisy chain*, the shift out from chip number n serving as shift in for chip number $n + 1$ and so on. The shift out of the fifth VA IC is not used as shift in for another VA, but can be transferred to the DTC. If there is a shift out coming out of the last VA after the correct number of clock cycles, the shift registers of all VAs work correctly and were properly initialized at the beginning of the readout sequence. As the VAs operate between -2 V and +2 V, the digital shift out is also a ± 2 V signal, which has to be transformed into a TTL signal. This is done by a fast digital comparator, whose output is connected to the SHO line. It was planned to as well include the SHO serializer shift register test into the power up system tests as to have the DTC DPR generate an IRQ if one readout sequence fails to provide a SHO after the right number of clock cycles. However, these features have not been

Channel	M_{11}	M_{12}	M_{21}	M_{22}
Bits	6,5,4, DTO	3,2,1, DOD	12,11,10, SHO	9,8,7, SST

Table 18: Assignment of ATC output bits to output channels. The numbers represent the fast ADC output bit with the corresponding number.

S_0	S_1	M_{11}	M_{12}	M_{21}	M_{22}
0	0	6	3	12	9
1	0	5	2	11	8
0	1	4	1	10	7
1	1	DTO	DOD	SHO	SST

Table 19: Assignment of ATC output bits to multiplexer control bit combinations.The numbers represent the fast ADC output bit with the corresponding number. The sequence given here for the multiplexer control bits is the same as applied during fast data acquisition.

implemented yet.

One 16 bit data word is transferred to the DTC for each clock cycle which is 200 ns long, so the multiplexer control bits have to be switched with 20 MHz. The output bits are assigned to the 2 multiplexer control bits, S_0 and S_1, and to the 4 data output channels M_{11}, M_{12}, M_{21} and M_{22} as shown in the tables 18 and 19. The assignment is the same for N- and P- compartment of the ATC. This is the order the DD circuit in MFPGA has to deal with. Please note that of the 16 data bits acquired the 4 MSBs are irrelevant in the way that they are not intended to transport relevant information. As 16 bit data words are stored in the module's FIFO, there is room for additional information.

8.9 The calibration chopper (CAC)

The calibration chopper circuit is meant to serve as generator for calibration pulses which are directly transferred to the hybrids. A calibration pulse simply consists of a voltage step fed into a calibration capacitor of defined size. This injects an amount of charge into the preamplifier channel selected with the serializer shift register when the hybrids are operated in test mode. The injected charge is

$$\Delta Q \;=\; \Delta U \cdot C_{cal}$$

with C_{cal} being the size of the calibration capacitor. The calibration pulse can be used to test and calibrate VA hybrids and to adjust the VA bias

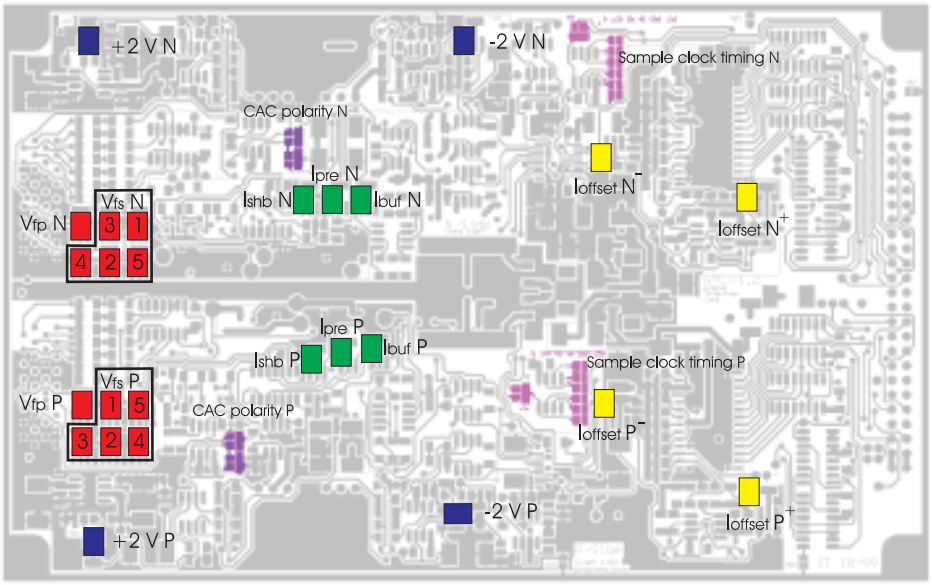


Figure 49: Location of all bias and offset potentiometers on the ATC card. Color encoding is as follows: *Blue:* Hybrid supply voltage adjust. *Red:* Hybrid bias voltage adjust. V_{fp} is common for all VAs, an adjust for V_{fs} is present once for every IC. The numbers given are the IC numbers on the hybrid the adjust belongs to. *Green:* Hybrid bias current adjust. A gauge for the bias current sources can be found in table 17. *Yellow:* I2UC offset current adjust. The offset current for each branch of the differential current output can be adjusted individually. *Dark violet:* Calibration chopper polarity jumpers. *Light violet:* Sample clock timing adjust jumpers.

parameters by directly observing the signal response to the calibration pulse at the hybrids analog output or behind the I2UC circuit. Moreover, the clock control circuit offers a feature to operate the fast ADC conversion clock without operating the VA hybrid shift clock, so the acquisition of single pulse shapes with 5 MHz sampling frequency is possible in principle. This operation mode together with the CAC would make an easy modification of bias parameters possible without the precautions usually necessary when observing analog signals directly on the ATC. However, this feature is not yet implemented, neither in software nor in the MFPGA programming[88].
The calibration chopper consists of a 10 bit DAC followed by a fast analog switch. The DAC value can be written serially with the SoSe. The fast analog switch essentially has 2 analog and 1 digital (TTL) input(s) and an analog

[88] An acquisition sequence without any DPR would be started, triggered by either the calibration pulse or an external trigger. The acquisition sequence would have an adjustable length and generate an IRQ after the sequence is finished, telling the software that a waveform is completely acquired.

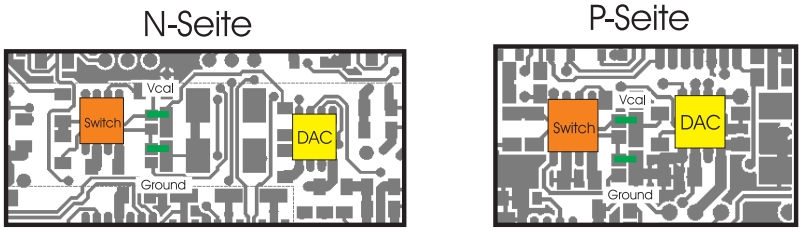


Figure 50: Calibration chopper layout showing the polarity jumper settings. One of the jumper pads is tied to ground, the other one to the calibration DAC output. Thus, the voltage step is always as large as the adjusted DAC voltage. Only the polarity remains to be set, i.e. the direction of the voltage step during a *strobe* low to high transition. N- and P-compartment have to process different polarities for real signals. However, the polarity setup is the same for both sides to ensure equal calibration signal polarities for the same *strobe* edge. Care has to be taken not to accidentally short the calibration voltage. Either the two upper (negative polarity) or the two lower (positive polarity) must be closed.

output. The TTL input of the switch can be operated using an ATC-DTC channel (*Strobe*), which is connected to a MFPGA input. It can either be operated with the SoSe or the MSQ (MSQ operation of the strobe signal is not implemented in the MFPGA programming yet). The state of the *strobe* input decides whether the voltage applied at input 0 or the voltage applied to input 1 is transferred to the output. One of the inputs is tied to ground. In this way, the voltage step applied between the switch inputs is always the DAC voltage set. The polarity of the voltage step during a low-to-high transition of the *strobe* input can be set using 2 jumpers right in front of the analog switch (see fig. 50). The default setting in the current telescope setup is the negative polarity for both ATC compartments.
The output of the analog switch is not directly used to feed the calibration capacitor, as the capacitance of the capacitor is small. The capacitor is integrated directly on the VA 2 hybrid and formed by 2 conductive plates in parallel separated by some dielectric. The capacitance of this arrangement is ≈ 0.5 pF. To adapt the DAC output to the dynamic range of the VA 2 IC, a calibration resistor of $R_{cal} = 6.4k\Omega$ resistor forms together with the 50 Ω termination resistor a 1:128 voltage divider. In this setup, a calibration voltage step of 1 V injects a charge in the preamplifier according to

$$\Delta Q_{inj} = \frac{R_{term}}{R_{term} + R_{cal}} \cdot C_{cal} \cdot \Delta U_{cal}$$

of $\approx$ 24000 electrons, which is a bit more that the charge deposited by a minimal ionizing particle passing through a 300 μm thick silicon detector[89]. As the linear dynamic range of the VA ICs is specified with $\pm$4 MIP and the output voltage range of the DAC is 0 to 4096 mV, the whole linear range of the ICs can be covered with the calibration chopper.
To test the CAC, first the DAC has to be programmed with the desired calibration voltage. Next, the serializer shift register of the VA hybrid has to be initialized and clocked until the desired channel is reached. As this cannot be done with the MSQ, it has to be done step- by step with the SoSe. To inject pulses, the *strobe* signal has to be operated. As described, the polarity of the injected pulse depends on the direction of the *strobe* transition. When operating the CAC with a complete hybrid assembly connected, make sure that the detector is depleted (while, of course, preserving the precautions recommended above). Admittedly, even when the detector is depleted, the hybrid channel which is injected in might start to oscillate. It is not yet fully clarified, under which conditions this situation occurs.

8.10 Monitoring circuits (MOC)

The monitoring circuits allows a software controlled measurement of certain ATC parameters. This is important especially for system tests and trouble shooting. After successfully performing the power up tests of the DTC, for example, the monitoring circuit can measure all analog parameters and the software can compare them to the values from the last operation period, which can be stored in an initialization file. This operation should be part of the standard power up procedure but is not yet implemented.
The most important part of the MOC is a slow 10 bit 8 channel ADC with an input voltage range from 0 to 4.096 volt. It can be programmed and read out serially. Each of the 8 inputs is connected to a specific voltage signal related to an important system parameter, which is to be monitored. The following parameters can be observed using the MOC:

- Channel 1: V_{cal}. Calibration voltage programmed into the calibration DAC. This is useful to check if both calibration DAC and MOC itself work correctly. It is also a check if the communication channels between ATC and DTC are operational. The DAC output is fed directly to the ADC.

- Channel 2: U_{th}. Thermistor voltage. Each hybrid comes equipped with

[89] This "standard charge" is in the following referred to as MIP charge or simply MIP.

a thermistor, which has a resistance of 10 kΩ at 25°C and a negative temperature coefficient. The thermistor sits in the middle of the hybrid. It forms together with a 10 kΩ bias resistor a voltage divider, whose "middle" voltage is connected with channel 2 of MOC. This voltage should be reciprocal proportional to the hybrid temperature. This feature was implemented to be used if the hybrid is connected to a cooling system to monitor as well hybrid temperature on both of the module's hybrids as the correct operation of the cooling system. A gauge measurement of the temperature is still to be done, as the cooling feature of the hybrid assembly has not been used so far.

- Channel 3: $+2V$. The positive supply voltage of the VA2 hybrid controlled by that ATC compartment. Indicates if, for what reason ever (shorts etc.) the voltage breaks down. It is directly fed to the ADC input.

- Channel 4: VI_{+}. The current draw of the positive supply voltage of the VA2 hybrid controlled by that ATC compartment. The current draw is measured by measuring the voltage drop over a small precision resistor R_m using an instrumentation amplifier with appropriate gain. The gain can be set by equipping the instrumentation amplifier with a "gain resistor" R_{gain}. The current draw for the positive supply voltage is $\approx$ 150 mA and R_m for the positive supply is $R_m = 0.22\Omega$, which yields a voltage drop of 33 mV. The instrumentation amplifier gain has been set to a value of 40 ($R_{gain} = 2.4k\Omega$) and this results in a measured voltage difference of 1320 mV at the instrumentation amplifier output. This voltage matches the input voltage range of the ADC and is connected directly to its input channel.

- Channel 5: $-2V$. The negative supply voltage of the VA2 hybrid controlled by that ATC compartment. Indicates if, for what reason ever (shorts etc.), the voltage breaks down. As the ADC can only sample positive voltages, the negative supply voltage is fed into an inverting amplifier with a gain of 1. The remaining OpAmp left in the quad OpAmp package used for generating the bias currents is taken for this purpose. The output of the analog inverter is fed into the ADC.

- Channel 6: VI_{-}. The current draw of the negative supply voltage of the VA2 hybrid controlled by that ATC compartment. The expected current draw is larger than for the positive supply, so the gain of the corresponding instrumentation amplifier has been set to 10 ($R_{gain} = 11k\Omega$). An expected current draw of $\approx$ 450 mA yields a measured

Data bus bit	Assignment	Contents for $S_0 = S_1 = 1$
D_0	M_{11} N	DTO N
D_1	M_{12} N	DOD N
D_2	M_{21} N	SHO N
D_3	M_{22} N	SST N
D_4	M_{11} P	DTO P
D_5	M_{12} P	DOD P
D_6	M_{21} P	SHO P
D_7	M_{22} P	SST P

Table 20: Assignment of ATC output data to data bus bits during SoSe controlled data readback. Bit 0 is the LSB and bit 7 the MSB of the data word read back.

voltage difference of 990 mA (with $R_m = 0.22\Omega$). As the ADC can only sample positive voltages, a voltage drop inverted with respect to the voltage drop of the positive supply is measured.

- Channel 7: $VPRB$. The bias voltage for the preamplifier bias current source of the hybrid controlled by that ATC compartment.
- Channel 8: $VBUB$. The bias voltage for the preamplifier bias current source of the hybrid controlled by that ATC compartment.

The ADC is addressed by writing a certain command word into its input shift register, telling the ADC which channel is to be converted and which conversion mode is to be used. Details about this can be taken from the ADC datasheet. After writing the command word, the data output multiplexer control bits should be switched to $S_0 = S_1 = 1$ to allow access to SST and DTO. The ATC data output can be read for N- and P- compartment simultaneously by using MFPGA mode address 21, encoding of the data output is as given in table 20. The SST signal indicates when the ADC has finished a conversion according to the last command word received. When the ADC is operated in internal clock mode[90], the SST moves from high to low after the command word has been fully received and moves back from low to high again after the conversion is completed[91]. Now the data can be

[90] This is recommended. The conversion mode is part of the command word. Please refer to the ADC data sheet for more details.

[91] A conversion takes ≈ 10 μs, so when checking for the SST one should not rely on finding the SST low before it goes back to high again. The conversion can be already completed before performing the first read byte access.

read back. Data is clocked out of the serial data out shift register of the ADC with the rising edge of a clock pulse.
Readback is done by subsequently giving clock pulses with the SoSe and performing a read byte access to MA 21, throwing away the irrelevant bits and putting together the complete 10 bit digital information.

8.11 The clock control (CLC)

On each ATC compartment, there are 4 devices which receive a clock signal from the DTC. As the number of signal channels is limited, it was decided to make the different devices share the same clock channel and send the clock signal through a clock control block. The clock control block decides which devices are going to receive a clock signal in consideration of two control bits, *CTR 1* and *CTR 2*, which can only be programmed using the SoSe. Each combination of control bits is called a *clock mode*. The control bits also control the state of the chip celects of slow ADC and DAC. The following four clock modes are available:

- *DAC:* The clock is distributed to the DAC only. This clock mode is to be used if the DAC is to be programmed or the DACs serial shift register is to be read back. The slow ADC chip select is inactive, the DAC chip select is active.
- *SLOW ADC:* The clock is distributed to the monitoring ADC. This mode has to be used for writing a command word into the ADC or reading back a data word from the ADC after the ADC has finished a conversion. The DAC chip select is inactive, the slow ADC chip select is active.
- *FAST ADC:* The clock is distributed to the fast ADC only. This mode is useful especially in combination with the calibration chopper to adjust bias parameters of the VA hybrids. After giving a calibration pulse, the MSQ can start an acquisition sequence, which samples the pulse response of the VA IC channel currently selected[92]. As described in the section about the calibration chopper, this feature is not yet fully implemented. The chip selects for slow ADC and DAC are inactive.

[92] One does not necessarily have to use the calibration chopper. Any device suitable for generating a VA response pulse and a trigger signal (e.g. a laser pulse generator) serves the same purpose.

Clock mode	*CTR 0*	*CTR 1*
FULL CLOCK	0	1
FAST ADC	1	1
SLOW ADC	0	0
DAC	1	0

Table 21: Assignment of ATC clock modes to control bit patterns.

- *FULL CLOCK:* This clock mode is the standard mode for testbeam data acquisition. Using this mode, the signal from the clock channel is fed to the hybrid and the same signal, delayed by the time specified with the DGS delay jumpers, serves as conversion clock for the fast ADC. The chip selects for slow ADC and DAC are inactive.

The assignment of the control bit combinations to the to the clock modes listed here is shown in table 21.

Figure 51: Photo of a module's power supply type NTBONN4. The only item on the front cover plate is the LED indicating system operation. The case size is 25 × 15 × 30 cm^3. The cases can easily be stacked.

Supply voltage	230 V AC, 50 Hz, 100 VA (max.)
Output 1 (low voltage)	2 bipolar and 1 unipolar voltages range 4.5 V - 7 V, 1.2 A
Output 2 (HV bias)	high voltage 80-180 V (default) , 1 mA adjustable between 65 V and 200 V

Table 22: Specifications of power supply NTBONN4.

9 Power supplies

As said before, each module has its own power supply. The power supplies, type NTBONN4, were developed by *GSP Gesellschaft für System und Produktentwicklung mbH* in Bad Dürkheim in Germany especially for this purpose. This firm previously developed low noise power supplies for various detector systems based on semiconductor sensor devices. The power supplies showed excellent noise performance and have proven to work very reliably. A photo of a power supply is shown in figure 51. An overview about the power supply specifications is given in table 22. Circuit details can be taken from the power supply section from the documentation folder.

Each module needs two ± 5 V symmetric power supplies, one for each ATC compartment, and a +5 V power supply for the DTC. The NTBONN4 power supply provides these voltages and an additional HV bias voltage for the detector itself. All these voltages are galvanically decoupled; any connections

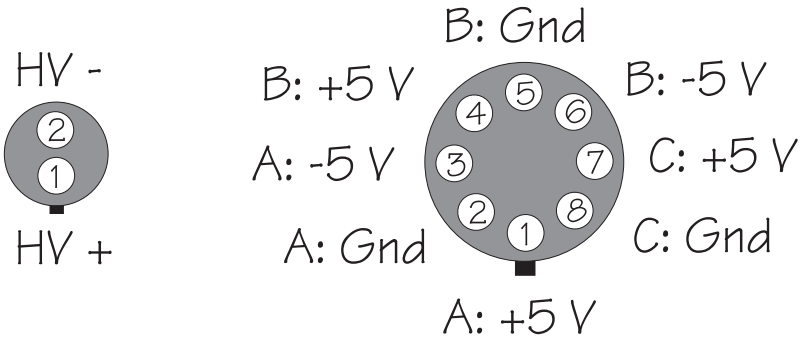


Figure 52: Assignment of output voltages to power supply output pins.

between the voltages are made inside the modules. How these connections look like during normal telescope operation is displayed in fig. 43. In the following, the power supply for the ATC P compartment is the A power supply, the power supply for the ATC N compartment is the B power supply and the DTC supply is the C power supply. It turned out that it is a good idea to operate as well DTC as both ATC compartments at $\pm$ 5.3 V and +5.3 V respectively. The exact output voltages depend on internal power supply settings. Inside the power supply, one potentiometer for each output voltage can be used to tune these voltages to any desired value within the specified range[93].

Figure 54 shows the back side of a power supply. The 2 pol. LEMO connector is the terminal for the HV bias cable. The HV bias voltage can be adjusted using the potentiometer. The HV variation range depends of the internal power supply settings given by the resistance of two additional potentiometers[94]. The HV bias voltage can be turned on or off independent from the low voltage power supply with an extra switch sitting directly under the HV adjustment potentiometer. If this switch is turned off, the bias voltage is shortened. This feature is useful when debugging the hybrid assembly or testing an ATC without a hybrid assembly connected. The large, 8 pol. LEMO connector is the low voltage supply terminal. Each low voltage power supply is fused with a fast 2 A fuse. The assignment of the fuses to the voltages is as follows:

[93]Each A,B and C voltage is generated by an individual card, three of which are contained in the power supply. Each card holds two potentiometers; one for the negative and one for the positive branch. For the C voltage source, only the positive branch is equipped with parts.

[94]The potentiometers for the adjustment of the HV bias range are sitting at the side of the case.

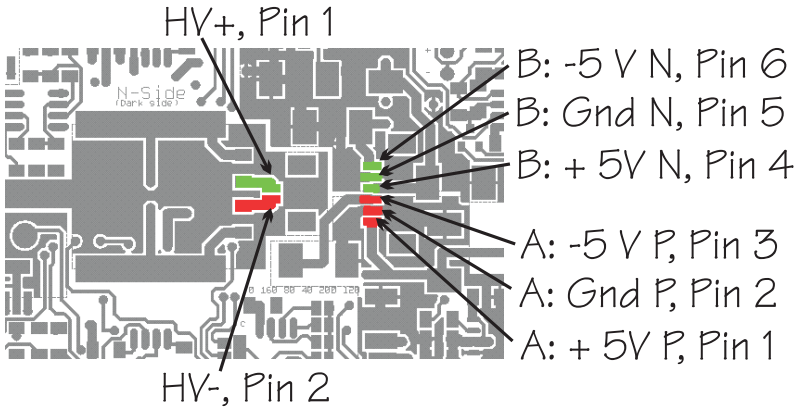


Figure 53: Connection between power supply and ATC. Green: Connections to the N-side. Red: Connections to the P-side. The location of the connectors is right in the middle of the ATC next to the first low pass capacitor. The DTC power is not connected to the ATC. Its connection is made directly to the back side of the DTC.

A: +5 V	A: -5 V	B: +5 V	B: -5 V	C: +5 V
F 1	F 2	F 3	F 4	F 5

The pin assignment of the different voltages to their connectors is shown in figure 52. Please note, that the assignment given here is also exactly the pin assignment on the *back side* of the associated connector in the corresponding module's case. Figure 53 shows, how the power supply output voltages have to be connected to the ATC. **Security information:** The power supply is only suitable for laboratory operation with all appropriate security precautions taken. Especially as far as the HV connector is concerned, care has to be taken not to touch it accidentally, as it is not protected against touch in a sufficient way. Large smoothing capacitors may cause large discharge currents although the current source itself is limited to 1 mA. Only open the power supply case after separating it from the power lines.

Figure 54: Rear view of a module's power supply. from left to right and from top to bottom: HV bias adjustment potentiometer, HV bias switch, HV bias output connector. Low voltage fuses, low voltage output connector. Line switch, power cord connector and line fuses are sitting together in the rightmost mounting.

10 The TLU

In the module setup described so far all modules act independently, their behaviour only depending on how the individual module parameters and preprocessor flags are set. But a telescope setup which is to acquire valuable data has to work coordinated, that means that it has to be made sure, that all modules take data for the *same* event. So there has to be some kind of trigger logic, a device, a piece of electronics, which makes sure that

1. all modules receive the same trigger signal belonging to the same event,
2. all modules *accept* this trigger signal, that means, all modules are able to process the event belonging to this trigger signal and really start processing.
3. Also, such a device is to be controlled by the DAQ PC, thus making a PC controlled run start or run abort possible.

Of course, these conditions, which are sometimes referred to as "trigger track proof operation" are not only to be met for the BAT modules but also by any kind of DUT connected to the telescope system. This is the weak spot of the system, the price one has to pay for the modules capability to act almost independently from any external DAQ control.
For this task, a special card has been developed, the *trigger logic unit*, which in the following will be referred to as *TLU*. In the following section, module means all kinds of modules (and not necessary BB modules), unless there is a special identification, e.g. DUTs and BAT modules.

10.1 PCB features

The TLU is a two layer PCB which is based on the DTC design as far as the connection to the BB is concerned. A view of the card is presented in figure 55. The TLU needs a voltage supply of 5 V - 5.5 V and has a current draw of ca. 300 mA. The central FPGA (CFPGA) of the TLU is a Xilinx XC3142, the same type which was used as IFPGA on the DTC. On the TLU, it not only holds the BB access logic (which is essentially the same as in the modules' IFPGA), but also a mode decoder, an interrupt status register and the trigger logic block. The functionality of these blocks will be explained in the section about the CFPGA. The driver logic, BB address jumpers and the CFPGA configuration section is the same as on the DTC. So, for example, if the CFPGA is configured in serial master mode, one has to provide a configuration PROM, plug it in the configuration PROM socket and open the jumper located right next to the top right corner of the CPFGA (holding the PCB with the BB connection up). If one wants to configure the CFPGA serially in slave mode via a download cable, one has to remove the configuration PROM and close the jumper.
Once again, the system clock comes from a 40 MHz crystal oscillator. There are not much more components on this card. There are two connector's rows, one input connector's row (ICR) and one output connector's row (OCR), each providing connectors for eight input output channels. The connectors are standard LEMO plugs. For the ICR, there are two connectors provided for one ICR channel, as, depending on the signal source, a termination resistor might be possible. Each input channel is also equipped with a 2 kΩ pulldown resistor. The ICR signals are fed into a fast eight channel driver, whose outputs are directly connected to the CFPGA. For the OCR, the CFPGA output signals are fed into a fast 8 channel line driver, whose outputs are connected via a 50 Ω series resistor to the OCR connectors. Care has to be taken here not to accidentally feed the TLU OCR output signals into an

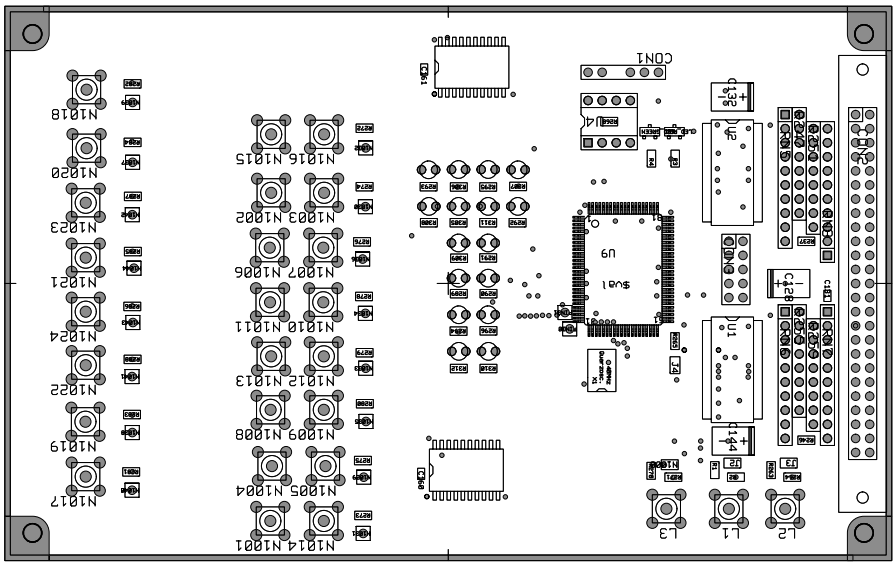


Figure 55: Layout of the TLU card showing component locations. The BB connector is on the right, input and output connectors row are on the left. CFPGA and indicator LED are in the middle.

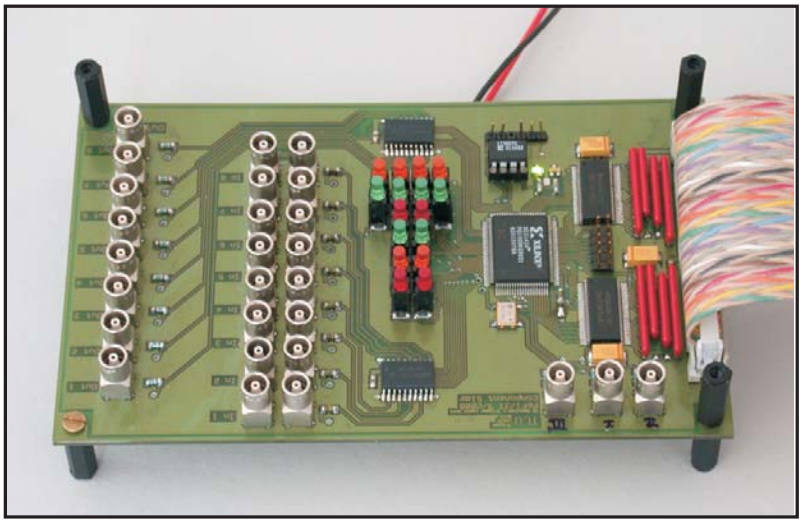


Figure 56: Photo of the TLU card. The allocation of the components is the same as in fig. 55.

input with an input impedance of 50 Ω. The outputs are set up for serial termination[95] , so connect them only to high impedance trigger inputs or, if

[95]Serial termination makes use of the fact, that the artificial output impedance (here the additional resistor) and the cable impedance of the signal line make up a 1:1 voltage

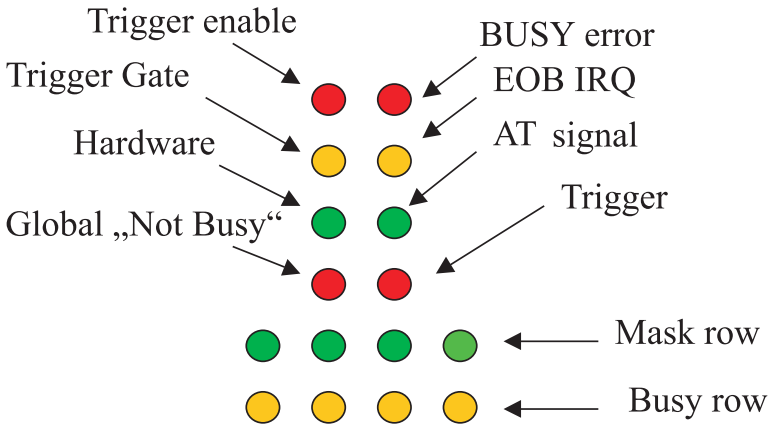


Figure 57: Order and assignment of indicator LEDs on the TLU. The indicator LED are meant to be visible from outside a testbeam area, indicating some important system status flags. Thus, the TLU should be placed in a location where it is observable.

necessary, use the OCR trigger output as input to a line driver with a high impedance input, which is capable of driving 50 Ω.
Between the CFPGA and the ICR, the indicator section (IS)[96] has been installed. The IS is made up of an arrangement of 16 large LEDs indicating the status of several internal and external signals of the CFPGA. The LEDs are large, because the TLU is meant to sit inside the beam area, but visible from the outside, so the user can keep himself informed about the system's state by simply looking at the IS LEDs. The meaning of the different IS LED has changed during various versions of the TLU firmware. The current setup is shown in figure 57.

Besides the IS LED, there are 2 SMD LED mounted at the same position as the first two DTC LED are and having the same functions: The green LED indicates proper configuration of the CFPGA, and the red (or yellow) LED indicates an IRQ generated by the TLU.
The resistors terminating the BB do not have sockets on the TLU, but are soldered directly on the PCB. This is because when there are one ore more modules being operated together with the TLU, there has to be an extraordinary module which alone has termination resistors. This is meant to be the TLU in all cases. Thus, as the TLU is the card which is always terminating the BB, there is no need to take the resistors out, so a socket is not needed.

divider. As the signal is reflected at an open end, the high impedance trigger input of the module or DUT, a signal with the correct voltage level will show up there.

[96]The IS is sometimes referred to as ACT (American Christmas tree), because of its arrangement and the many different LED colours.

To remind the user of the telescope of the fact that

1. a TLU (and only one) has to be present to make the modules work correctly and coordinated together,
2. a TLU has to be present to terminate the BB,
3. none of the other modules must carry termination resistors on it and
4. no modules are to be connected to the BB behind the TLU (so it it the TLU which not only terminates but also ends the BB),

the TLU BB address has to be 7[97]. In principle, one can think of a scenario where more TLUs are daisy chained, e.g. if there are a lot of DUT modules whose readout has to be coordinated, but one would rather use some separate logic extensions (e.g. NIM modules) in this case. For normal use, the logic provided by the TLU firmware, will be sufficient.
Besides ICR and OCR, the TLU provides three additional LEMO ports. These LEMO ports are the equivalent to trigger input, inhibit input and busy output of the DTC. Their occupation with signals again depends on the TLU firmware revision.

10.2 Principle of operation

The TLU is the trigger controlling stage of the telescope. It also terminates the BB, therefore the DAQ software enforces the presence of a TLU. The TLU processes the "MODULE BUSY" (MB) signal of each module, so each module's MB is to be connected to one input of the ICR. The TLU is configured at the beginning of each run with a configuration depending on which modules are to be read out, thus, which of the ICR ports are to be regarded while making the trigger decision. there is a convention about this. The ICR port a module's MB is to be connected to corresponds to the module's BB bus address. If this demand is not met, confusion might result e.g. in the case that one module is taken out of the readout. If there are any DUTs connected to the system which also create a BUSY signal (e.g. due to some deadtime), care has to be taken here to tell the TLU which ICR port corresponds to the DUT busy. This has to be done correctly; otherwise trigger confusion might mess up the whole DAQ run. ICR port 8, by the way, is occupied. The connection of the trigger coincidence to the TLU has to be made by connecting the trigger input to ICR port 8. Please note that as long

[97] This is demanded by the DAQ software package.

as the TLU-distributed trigger signal is used, a DUT must only generate a BUSY, if its overall deadtime, the time it is not sensitive to trigger signals, exceeds the minimal dead time of a telescope module and / or there is some additional dead time during data readback, buffer overflow (if it is operated in buffered mode) or so. But if the deadtime of a read out device is very small or if the device supports "trigger pipelining", care has to be taken. it takes some time for the modules to recognize the trigger signal, to start processing the event and to generate their MB, and in the time scales we are talking about cable delay have to be taken into account. To prevent an additional trigger slipping by during the small time window between the trigger signal and the arrival of the MBs, an *artificial timeout* (AT) has been implemented. The TLU won't let any triggers pass by for 6.4 μs after the rising edge of the last valid trigger signal. At this time, even the last and slowest module should have sent its MB. This timeout does not affect the telescope readout, triggering or speed, for the telescope modules can only process one event at a time and the MSQ runtime is about two orders of magnitude longer; it is only meant to ease the operation of additional modules which support multiple triggers and stuff.
The TLU generates a gate signal from the MBs (and the AT). Only signals of modules which are to be read out have to be evaluated while generating the gate signal. This gate signal is active if all modules are not busy. Only if the gate signal is active, the TLU generates a module trigger in reaction of an external trigger input[98]; otherwise the trigger input will be ignored. As in most cases the system will be operated with 4 telescope modules (which will probably have BB addresses 1 to 4), the first 4 of the ICR ports are needed for the telescope modules. The remaining slots can be used as input for busy signals of devices under test or other trigger controlling stages. As mentioned, the external trigger input has to be connected to slot 8 of the input connectors row[99]. A connection example for the TLU is given in figure 58.

The "normalized" trigger signal produced by the TLU with the present firmware version if the trigger condition is met (which means that all connected modules are not busy and a trigger coincidence occurs) has the following properties:

- The trigger pulse is active high,
- it has TTL levels,

[98] Which comes, in most cases, from an external trigger coincidence.

[99] In the version of BAT DAQ used for testbeam operation in July, August and September 2001 ICR slot 7 for example, was occupied with the busy of a VME-TDC.

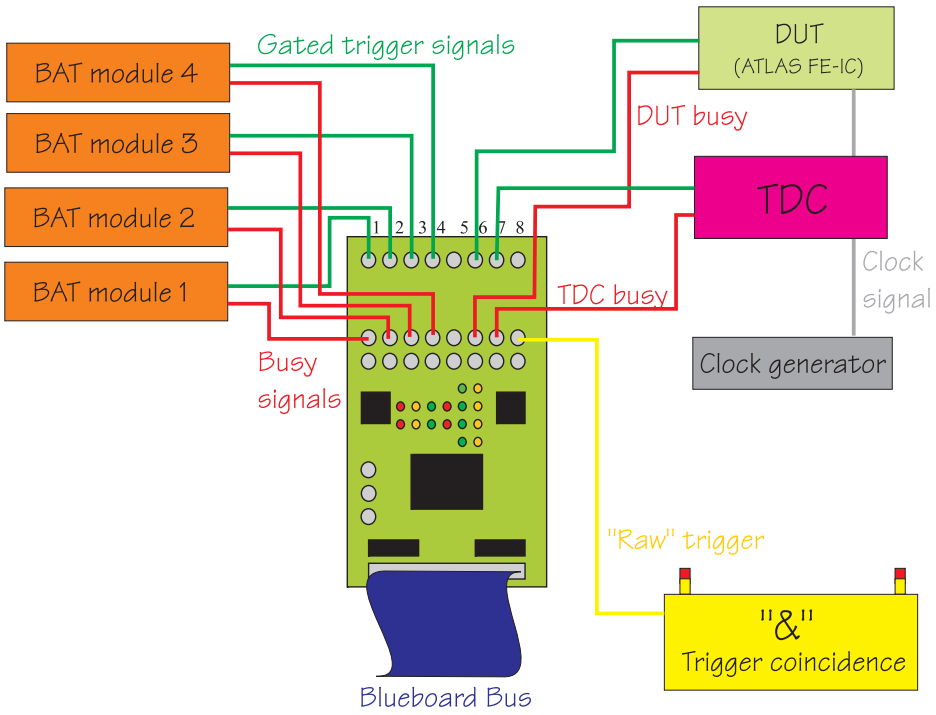


Figure 58: Connection example for the TLU. An example for 4 modules, 1 TDC and 1 DUT is shown. Here, care has to be taken to provide the DUT the same clock as the TLU, otherwise the correlation between trigger and DUT clock is lost. All the trigger signals are not synchronized with the TLU clock on their rising edge. The example is shown for a FE-DUT module bearing BB address 6. Input 7 has been chosen as TDC busy input by convention in the BAT DAQ. A PLL DUT does not need a busy signal as the PLL has approximately no deadtime.

- it is 100-125 ns long with
- the rising edge being asynchronous,
- the falling edge being synchronized with the TLU's 40 MHz crystal oscillator.

The most important thing here is that the rising edge of the trigger is not synchronized with the 40 MHz clock, so that the trigger signal is delayed by a fixed delay relative to the original trigger coincidence. So, the trigger signal still carries the "real-time" timing information with a jitter much smaller than 25 ns. This is not that important for telescope operation, but as soon as DUT are connected, for which precise timing information is important (e.g. ATLAS pixel front end ICs), this has to be taken into account.

The three additional LEMOs are, in the TLU firmware version described here, occupied as follows:

1. LEMO 1: Output of the trigger gate signal. This signal can serve e.g. as a gate input to some extended trigger logic. The gate is active high.

2. LEMO 2: Output of the gated trigger signal, simply the trigger signal input sent through a logic AND together with the trigger gate. The shape (length etc.) of the output signal depends on the kind of trigger signal fed into the TLU.

3. LEMO 3: End of burst: As will be described in the section about TLU IRQs, the TLU is capable of generating IRQs. One of these IRQs is generated as reaction to a certain kind of input signal, the End-Of-burst signal. More on that later.

The TLU offers two possibilities to generate triggers independent from any external trigger input sources. First, the TLU enables the user to create a trigger as a reaction to writing a certain MA, which is operated in strobe mode. The trigger pulse generated in this way has exactly the same properties as the trigger pulse generated from a normal trigger input. Second, the TLU owns a hardware trigger generator based on a 16 bit counter, which is clocked with $\frac{1}{16}$ of the 40 MHz system clock. A register controlled multiplexer enables the user to select one of the counters 8 MSBs as a trigger signal input, thus making the TLU generating triggers at certain fixed frequencies between $\approx$ 5 kHz and $\approx$ 40 Hz. It is planned to enable the TLU to generate random trigger pulses with a certain mean frequency for more realistic test of the DAQ chain. This can be done e.g. with a linear feedback shift register or a similar technique. This feature will be implemented in a later version of the TLU firmware.
Both features can be used to debug the whole data acquisition chain, especially after integrating a new DUT into the readout. In combination with some dummy data generation algorithm data volume storage, online monitoring and speed tests can be performed.
It was said before that the most important task of the TLU is to enforce synchronous triggers. When doing this, there are two things one can do:

1. One can evaluate the MB signals before making the trigger decision to ensure that all modules can really process a given trigger.

2. One can additionally check after each trigger if all modules accepted the given trigger signal.

The TLU fulfills the first task by generating the trigger gate with the properties described above. The second task is especially important while operating the telescope in DIRQ mode, for then it is not that easy to find out if a module didn't take data for a certain trigger event, as the event data arrives in bursts, uncorrelated to the time the data was actually taken. To check if a module accepted a trigger signal, the TLU relies on the MB's. The TLU checks, if, after a certain time after the trigger signal, a modules MB had a low to high transition. It is not sufficient to check for the MB being active, for a fast DUT module for example might have finished processing by this time. The TLU uses the finishing of the AT (6.4 μs after the trigger occurred) to perform this check. If the result is positive, nothing happens. If the result is negative, that means, one module didn't send a MB although it was expected to do so, the TLU can (if wanted) generate an error IRQ. This is important, for, if such a condition occurs, something went terribly wrong.

10.3 Implementation

A block diagram of the CFPGA implementation is shown in fig. 59. The CFPGA contains, just like the IFPGA of the modules, an 8 bit mode address register and a mode decoder. The 8 bit address space is not fully used, only 7 MA are used.

10.3.1 Mode addresses

Please find below a complete description of the mode address registers used in the TLU.

1. *MA 0: SEL RDID* (read). Readback of the TLU firmware implementation ID. in the current TLU revision this ID is 1 (second version), the class ID of the TLU is 4.

2. *MA 1: SEL ISR* (read/write). The interrupt status register of the TLU works exactly as the ISR of the BAT modules. MA 1 can be used to access it.

3. *MA 2: RW REG* (read/write). RW REG can be used to access the CMD register of the TLU (there is only one CMD register here).

4. *MA 3:* unused.

5. *MA 4:* unused.

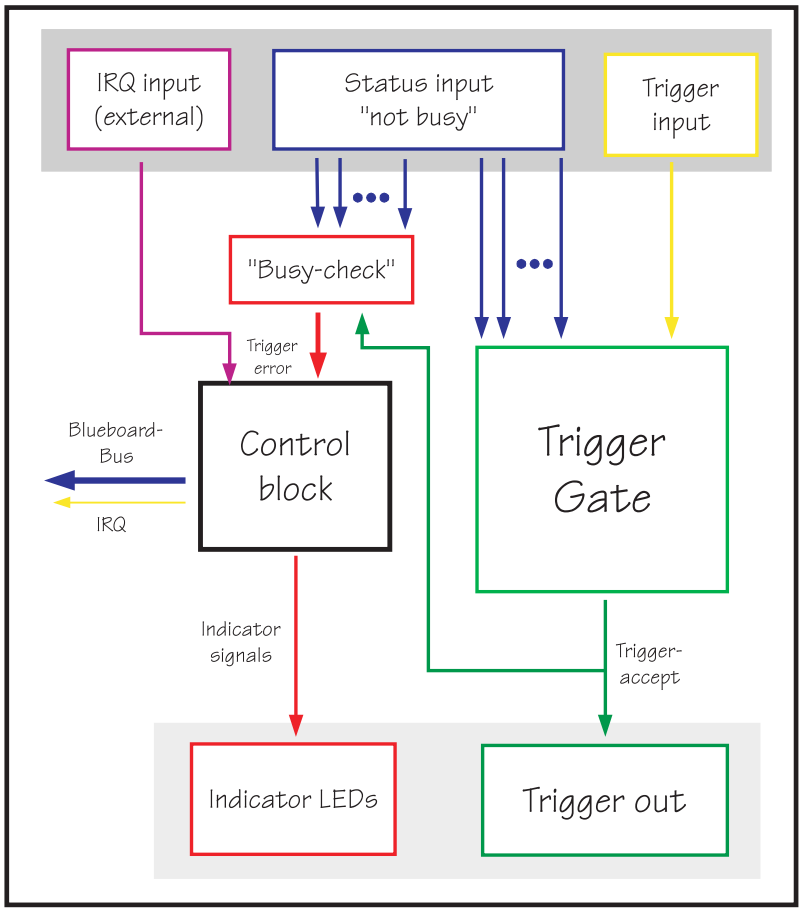


Figure 59: Block diagram of TLU CFPGA implementation. The top row (dark grey) are the different LEMO inputs. The *trigger accept* is generated depending on the state of the status signals and the state of the trigger input. The trigger accept also arms the *busy check*. This block expects a busy signal from every trigger receiving module certain time after the trigger occurred.

6. *MA 5: READ ICR* (read). Reads the current status of the ICR bus. Can be used for debugging (e.g. if one module gets hooked up)

7. *MA 6: READ OCR* (read). Reads the current status of the OCR bus. Mostly unused feature.

8. *MA 7: RW MASK* (read/write). Used to access the trigger gate mask register.

9. *MA 8: SOFT TRIG* (strobe). If the module is operated in soft trigger mode, an access to MA 8 generates a "normalized" trigger pulse.

10.4 The CMD register

The CMD register is an 8 bit register. All 8 bits are used.

1. *CMD bit 0: Enable IS.* This bit, if set, enables the TLU IS. The indicators won't be active otherwise.

2. *CMD bit 1: Enable Trigger.* This bit has to be set in one wants the TLU to generate any trigger signals at all. This is the signal, which is set or reset if one wants to start or end a run with the DAQ software. All other bits are initialized before.

3. *CMD bit 2: Select Int / Ext trigger.* This bit selects the source of the trigger edge. If the bit is set, the trigger edge will come from the ICR port 8 (EXT TRIG), else, the TLU will take the trigger edge generated by the internal trigger source (INT TRIG). In any case, bit 1 has to be set.

4. *CMD bit 3: Enable HARD TRIG.* If this bit is set, the hardware trigger sources (INT or EXT) generate the trigger edge. Else, the trigger edge can be generated with MA 8.

5. *CMD bit 4: Enable INT TRIG.* This bit enables the internal trigger source. It has to be set in addition to bit 3 and bit 1 to have TLU internal triggers generated. Bit 2 mustn't be set then.

6. *CMD bit 5-7: Internal trigger speed.* This control group allows a selection of the frequency with which the internal TLU trigger source generates triggers. The following values can be selected:

   (a) Value 0x00. Trigger speed $\approx$ 5 kHz.

   (b) Value 0x20. Trigger speed $\approx$ 2.5 kHz.

   (c) Value 0x40. Trigger speed $\approx$ 1.2 kHz.

   (d) Value 0x60. Trigger speed $\approx$ 600 Hz.

   (e) Value 0x80. Trigger speed $\approx$ 300 Hz.

   (f) Value 0xA0. Trigger speed $\approx$ 150 Hz.

   (g) Value 0xC0. Trigger speed $\approx$ 80 Hz.

   (h) Value 0xE0. Trigger speed $\approx$ 40 Hz.

   Please note that the real trigger speed may vary from the values given here, depending on the real frequency of the TLU's crystal oscillator.

10.5 The TLU mask register (TMR)

The TLU mask register is the most important register in the TLU. Its contents decide whether a modules MB is taken into account while generating the trigger gate or not. It also contains a global enable for test purposes. The TLU mask register is an 8 bit register. Each register cell corresponds to one ICR input slot, except for bit number 7. The MB of a module connected to a certain ICR slot is evaluated if the corresponding TMR bit is set to zero (!!!!!). If the bit is set to 1, the IRC slot will appear for the following gate logic as *being permanently not busy*[100]. Bit 7 has a special position. As ICR port 8 is occupied as input slot for the trigger input from the external trigger coincidence, bit 7 does not correspond to a certain ICR input slot and thus to a certain module. It is used as a global mask bit. If bit 7 is set, none of the other MB signals will be evaluated. This is a feature mainly used for test purposes of the mask logic. So care has to be taken not to set bit 7 accidentally to 1 during normal operation.
The TLU, and especially the TMR has to be initialized at the beginning of a new run. The mask bits of the modules which are connected to the system and which are to be read out have to be set to 0, and the global mask bit also has to be set to 0. Of course, the software initializing the TLU has to know which module is connected to which ICR slot. As long as there are modules connected to the BB, there is a convention which is easy to meet. A module has to be connected to the ICR slot, whose number corresponds to the module's BB address. BB address 0 is occupied by the BB-PCI card, and BB address 7 by the TLU itself. The BAT modules, which are in use at the moment, have the BB addresses 1 to 4, so the addresses 5 and 6 could be occupied either automatically by DUTs with the corresponding BB addresses, or, as long as no BB modules with those addresses are present, by other, non-BB-based DUTs. This then requires some software work.
It is unlikely that the system will be operated with more than 2 DUTs in normal operation. But if, for what reason ever, the "natural" resources of the TLU are not sufficient for safe operation, the user is enabled to set up additional gate logic by using the signal output to the three basic LEMO connectors next to the CFPGA.

[100] This is due to the fact that one wants to generate a *gate* signal form an active high *busy* signal.

10.5.1 TLU interrupts

In earlier versions of the TLU firmware, IRQ generation was not possible. This has changed due to the many benefits coming along with TLU IRQ handling. As the TLU ISR is set up identically to the BAT modules ISR, the mechanism of IRQ generation and handling will not be discussed here. Instead, the different IRQCs and their handling will be explained.
There are 4 IRQ addresses in the TLU, just like for the DTC IFPGA ISR. Of these 4 IRQAs, 3 are used.

IRQA 0: The trigger IRQ. This IRQ has been implemented to speed up DAQ especially in the case there are DUT modules connected which are neither capable of IRQ handling nor data buffering. So far, the BAT modules have been operated in EIRQ mode. Thus, an IRQ was generated after each event by all the modules, and the DAQ loop first processed the DUT modules data before processing the BAT module data. As an IRQ was generated for each event, no DUT data got lost. But as explained in the section about acquisition modes and sequencer runtime, the deadtime of the system increases, as the readout time for the BAT modules multiplies by the number of connected BAT modules, and the DUT readout time also has to be taken into account. A first step to speed the readout up a bit was to operate only one module in EIRQ and the other modules in DIRQ mode, so the total readout time consists of the readout time of only one BAT module and the DUTs, the other BAT modules being read out occasionally only[101]. But this is not a satisfying situation, because the BAT modules are only being operated in the slow EIRQ mode because the DUTs depend on an BB IRQ.
The solution is to introduce the TLU trigger IRQ. It is generated, whenever the TLU sends a trigger signal. As the data processing for any kind of DUT will probably take some time, one can select an adjustable delay between trigger occurrence and IRQ generation. The IRQ handling routine then can process the data from all DUT modules which rely on the event IRQ. The processing of the modules' IRQs runs independently in parallel. In this case the module has to be busy independent from the MB signals as long as the TLU trigger IRQ is not yet fully processed (but, of course, lasts longer if e.g. processing and data readback of the TLU IRQ depending modules does not take as long as the MSQ runtime). Adjustable IRQ delay and TLU IRQ busy are features not yet implemented in the current firmware revision (IID number 1), but will be implemented in the next revision.

[101] Whenever their individual IRQC is met.

IRQA 1: The EOB IRQ. Sometimes it might be useful to artificially generate IRQs as a reaction to special signals from the outside. The TLU has a dedicated LEMO port (LEMO 3) for this purpose. Whenever a TTL high pulse is applied to this port, an IRQ will be generated. This IRQ has no consequences inside the FPGA, it will not produce a busy or stuff. But it may be used to signal special conditions to the DAQ. The easiest example is the example which gave birth to this IRQ mode's name.
In the H8 testbeam location at CERN the test beam generated by the SPS comes along in spills each of which is approximately 5 seconds long. The whole cycle from injection over ramp-up to extraction takes ≈ 10 s. Between two spills there is a break of 5 seconds. A special *beam logic* provides a *start of burst* (SOB) and an *end of burst* signal (EOB). The EOB indicates the beginning of the 5 second break between two spills. The time between the spills can be used e.g. to empty all system FIFOs (especially those in the BAT module's FIFOs when running in buffered mode) and to compare trigger numbers after doing this to ensure that the triggers are still synchronous and eventually generate an error message if not. Afterwards, the readout has to be re-initialized to be able to cope with the next spill. The task described above could be taken by the handler of an EOB IRQ if the EOB signal provided by the beam logic is used to generate an IRQ. But of course one can think of many other reasons such a "customized" IRQ might come handy in.

IRQA 3: The error IRQ. There is only one thing that could possibly go wrong in the TLU. If the TLU sends a trigger signal to a certain number of modules, they have to "answer" the trigger signal with a "busy" condition, so the MBs of the modules connected should become active. If this hasn't happened for one module after some time, it is unlikely that it will ever happen, so the module missed this special trigger. In this case, trigger synchronity is lost, and the run is in danger to become messed up. The TLU performs an automatic "busy check" test. With the occurrence of a trigger a 7 bit register is cleared, each register corresponding to a ICR slot. The low to high transition of one MB sets the corresponding register to 1[102]. The result passes trough a logic OR gate with the slot's mask bit as second input. So this very bit is active, if either the MB became active yet or the mask bit being set indicates that this module is not to be regarded while evaluating

[102]Yes, of course, care has to be taken here not to accidentally clock the register with a spike on the MB line. This is all foreseen and conditions like these can hopefully be excluded. For implementation details please refer to the schematic of the corresponding TLU building block

the MBs. All seven register bits are fed into a 7-input logic NAND gate. The result serves as an input for a register cell which is also cleared by the trigger signal, but which is clocked once when the AT runs out. Thus, 6.4 μs after the trigger, this register contains the information whether all important modules answered the trigger with a busy or not[103]. And if the IRQ is enabled, the TLU will generate an Error IRQ indicating that something went terribly wrong.
Possible reasons for this error are mistakes while connecting the modules to the TLU (check all ICR slot numbers, not only for the BAT modules, but also for the TLU), missing trigger cables or missing MB cables. Also, if e.g. the TLU operation voltage (supposed to be between 5 and 5.5 V) is too low, a module might not be able to recognize the trigger signal. The same, if accidentally one module was turned off. If all this fails, you and your system, are in serious trouble.
Here, the description of TLU operation finishes. For more details, please consult the schematics. The TLU should provide all that is necessary for a complete an trigger track proof operation of the telescope. All that is missing at the moment is some solid case.

[103]The AT margin is selected for the same reason as it is selected for the AT: By this time even the slowest module should have noticed that such a thing like a trigger pulse recently occurred. If not, the design of the module should be revised.

11 Software issues

Of course, a description of the telescope software can not be as detailed as the hardware description. Many things are still in progress here, concerning the adaption of the system to different operating systems and programming languages. For this reason, a short description of *BATDAQ* is given. This is the software package developed in parallel to the BAT hardware in Bonn which was successfully used in several testbeam periods at the ELSA accelerator testbeam facility and, with some extensions, at the H8 testbeam at CERN. The package runs under Windows 9x / NT 4.0 and was written in C++ using the Borland C++ Builder (version 3) as development environment including TeeChart 4.0 for advanced plot and display capabilities. As the whole system was designed for PC based operation and at the time the development started only Windows drivers for the PCI interface card were available, choosing a Windows OS was natural under the circumstances.
For several reasons, at the moment some effort is made by members of the ATLAS collaboration to adapt the system to Linux OS by writing the appropriate device drivers for the hardware and re-writing parts of the BATDAQ package. Additionally, the device drivers for the VME-BB interface mentioned in the section about the BB are being developed, but this is not a subject of this description. The description is meant to provide a new user enough information about the code structure to make him explore and understand the code on his own.

11.1 General structure

Figure 60 gives an overview about the structure of the BATDAQ package. It consists of two to four processes, which run independently. Each process possesses its own GUI. All processes communicate with each other by exchanging windows messages. Data is sent between the processes via so-called *shared buffers*, OS administered memory blocks in system RAM to which, due to memory mapping, several processes can gain read and write access. The writer configures the system, controls the data acquisition to all modules connected to the DAQ readout chain and sends the events acquired in this way to the writer. The writer takes events belonging to the different modules and assembles module events from them, which are written to disk or sent to the different OMO processes on demand. The OMO processes allow the user an on-line overview about the system performance.

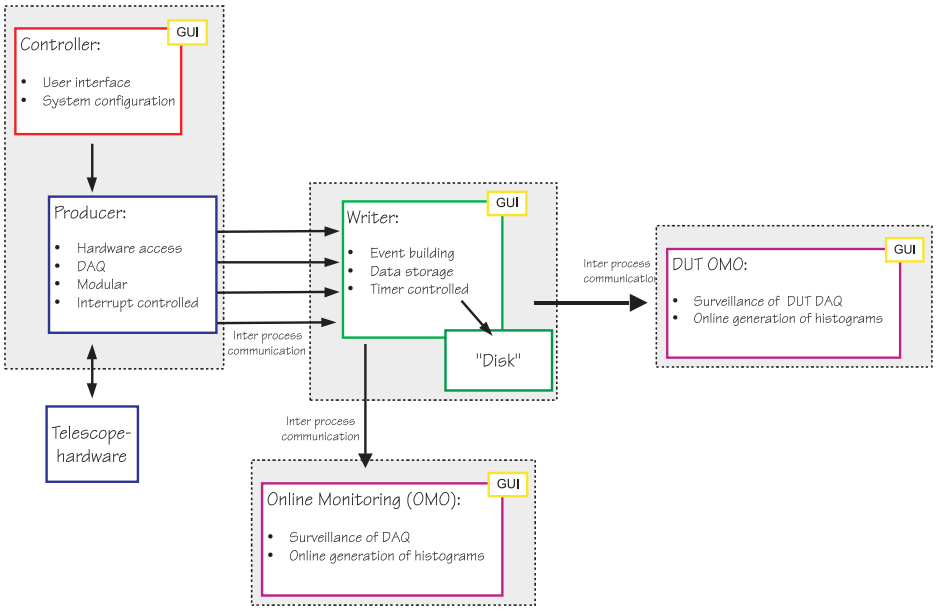


Figure 60: Structure of the BATDAQ software package. Up to four independent processes communicate via inter-process communication channels (shared buffers). Each process provides a separate GUI. The *producer* (PDC) controls the hardware and the DAQ, the *writer* (WRT) receives the data acquired by the PDC and is furthermore responsible for event building and coordinated storage of the data. The WRT also sends the data of selected events to the *online monitoring* (OMO) processes. In BATDAQ, separate OMO processes exist for BAT module data (BAT OMO) and DUT data (DUT OMO).

11.2 The producer (PDC)

The PDR is the central system configuration and DAQ process. It is the only BATDAQ process which directly accesses the hardware. The producer configures and administers the hardware and offers a flexible GUI for this purpose. The producer also owns the *run control*, DAQ runs are started and ended by the producer. It is also the process, which handles the different kinds of interrupts on the BB generated by the BAT modules, the TLU or some DUT. As most of the IRQs which are to be processed are interrupts indicating that there is some data to be read back, the producer is the very process, which in fact "produces" the data.

11.2.1 Compatibility

At the moment, the producer supports six different kinds of modules, which can be divided in two "classes":

1. *BB based modules.* BB based modules are connected directly to the BB. As described in the section about the BB, the PDC scans the entire BB address range during startup in search for connected modules. In case a module is found, the type of the module is recognized, the appropriate software modules (class instances) are generated and the GUI is automatically initialized accordingly. The following kinds of BB based modules are supported by BATDAQ:

   - *BAT modules*: This is the standard BAT telescope module configuration described above.
   - *TLU*: The TLU is the only module which is demanded by the software. The program will abort if during startup the absence of a TLU is detected.
   - *FE-DUTs*: The FE-DUTs were designed to operate ATLAS front end single chip devices BB based together with the BATDAQ and used in several testbeam periods. This is an example for easy integration of a DUT into the BAT software environment.

   The BAT modules and FE-DUTs are both based on the DTC interface card. For the FE-DUT modules, the DTC was equipped with a new analog interface to meet the requirements of an ATLAS front end IC and the main FPGA programming had to be changed. Although their type is automatically detected during startup, the main FPGAs of both FE-DUT and BAT module still have to be configured. This can be done either manually using the GUI or automatically with a configuration file.

2. *VME based modules*: Although the BAT was designed to operate completely independent from VME, the support for VME modules had to be added on demand for compatibility reasons. The support for VME modules is not optimal yet. Reasons are as well limited features on VME module side as unused features on software side. For all accesses to VME based modules, a common National Instruments PC-VME interface has been used.

   - *PLL ATLAS single chips*: The operation of ATLAS single chips under "official" testbeam conditions demands the use of a special digital interface card, the *Pixel low level card* (PLL). BATDAQ supports the readout of PLL based ATLAS pixel front-end single chip devices.
   - *PLL ATLAS modules*: ATLAS pixel front end modules can also be operated by the BATDAQ using the PLL.

- *CAEN TDC*: The support for a special type of CAEN TDC was implemented also for ATLAS testbeam use.

The main problem with the VME based modules was that VME IRQ support was not implemented. Although for the TDC a work-around had been found, for the PLL based modules only event-by-event readout was possible. Moreover, automatic detection of VME modules and GUI / software initialization is not implemented in the software. This can either be done manually or using a configuration file.

So the first thing, a user has to do, is to complete system setup manually or by loading a system configuration file.

11.2.2 Graphical user interface

The PDC offers a flexible GUI for initialization and configuration of the BAT modules, which allows

- the individual selection of MFPGA configuration files,
- initialization of the MSQ hold delay,
- pedestal taking with a variable number of pedestal events,
- threshold configuration,
- setting of the IRQ mode and
- the setting all DPR flags.
- Additionally, individual modules can be ad libitum taken out of or included in the readout.

The PDC also offers a versatile tool for viewing the pedestals of the modules with or without pedestal correction, giving also an overview about the system's noise and the common mode behaviour. Additionally, a hit bit display allows the identification of noisy channels after threshold setting. Also, the PDC GUI offers a lot of test facilities, from digital tests of the DTC card to analog readback of the ATC MOC and tests of individual channel's behaviour using the *Mäuseklavier*, a tool for the manual operation of the SOSE and the CAC. The software demands a digital system test after power up, sequencer initialization, pedestal acquisition and threshold setting for all BAT modules

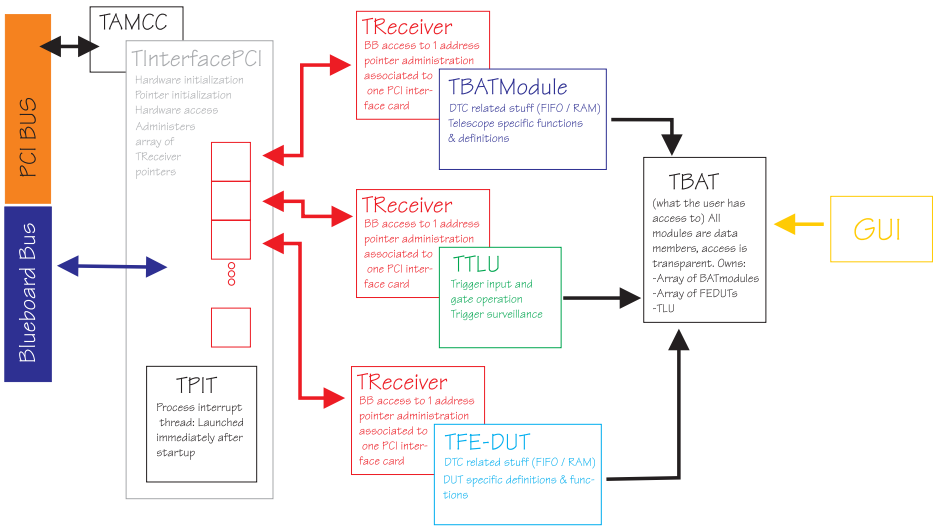


Figure 61: Class hierarchy of the producer core. The BB modules are all child classes of the `TReceiver` class. The `TInterfacePCI` class administers a list of all connected receivers. Each module has to register at the instance of `TInterface PCI`. Thus, multiple module addresses or missing modules can be prevented. There, the `BB pointers` are initialized. The class also owns the `TProcessInterrupt` thread, which is the genuine producer thread. The instance of the BB module classes is what the user has access to from the GUI.

included in the readout chain, before a run can be started.
For the TLU, a similar configuration tool exists, but with restricted capabilites. TLU IRQs which are to be activated can be selected, the trigger mode can be set and, if the internal hardware trigger generator has been selected, the trigger speed of the trigger generator can be set. For the FE-DUTs, a GUI similar to the BAT module GUI can be accessed, if a FE-DUT module is connected to the system.

11.2.3 Module administration

Due to the large variety of modules which can be connected to the BB, initialization and administration of the BB system is an important issue. This task is made easier by the class structure which is at the bottom of the software. An overview over this structure is tried to be given in figure 61. Every BB device is in software described by a special class. All of these classes are child classes of a single base class, `TReceiver`, which contains all BB communication specific functions and data members, particularly the pointers for BB access, the `BB pointers`, mentioned in the section about the BB. The different module specific functions are contained as extra members of the child

class. The PCI interface card itself is also described by a certain class, the class `TInterfacePCI`. This class is derived from the `TAMCC` class describing the PCI matchmaker IC and is responsible for the proper initialization of all hardware related stuff.
The user has access to instances of the classes describing the BB modules. But these classes have to be initialized via the instance of `TInterfacePCI`. When an instance on this class is generated, the hardware of the PCI interface is initialized first. Afterwards, the modules connected can be registered. This is done in the following way:

- The software can find out which BB modules are connected to which BB address by using the so-called `LookUpModules` function. Then, for each connected BB module an instance of the class describing to is to be generated.
- The `TInterfacePCI` class contains a list of pointers to `TReceiver` structures. Whenever a new BB module is generated, a pointer to the `TInterfacePCI` instance the module is connected to has to be given to the modules constructor[104] , as well as the BB address the module is carrying.
- The `TInterfacePCI` instance then checks, if a module bearing this address is connected to its BB[105]. If this is successful, the `TReceiver` pointer of this module is added to the interface's receiver list at the position corresponding to its address. The `BB pointers` are initialized afterwards.
- The instance of the BB module class, which is now initialized also has to know the pointer to the instance of `TInterfacePCI` it belongs to. So a pointer to `TInterfacePCI` is a data member of `TReceiver` and is initialized after the correct registration of the module in the interface's receiver list.
- If a BB module instance for a certain BB address is about to be generated without being connected to the BB, the interface will notice and pose an error message. When a BB module instance for a certain BB address which is already occupied, the interface will notice due to the pointer in the receiver list being already initialized and pose an error message.

[104] Each module has to know to which instance if `TInterfacePCI` it belongs, as more than one BB PCI interfaces can be operated in the same system at the same time.
[105] This is done in the way described in the BB section

This process takes place when generating the instances of the module classes or the instance of `TInterfacePCI` respectively. The software is not set up to handle the case that modules can be added or deleted from the module list with the program still running. Whenever a hardware change on the BB bus took place, when a BB module was removed or added, the program has to be restarted.

11.2.4 Run startup and shutdown.

The run control mainly consists of two buttons, a start run button and a stop run button. The number of events to be taken in the next run, the DAQ goal, can also be entered. A message window displays messages posed by the run initialization function.
When a run is to be started, each module which is to be included into the readout chain is initialized with the parameters and files specified before using the configuration GUI. Additionally, the IRQs for this module are activated. If the initialization of a module is successful, a shared buffer is created for this module in which later the IT will write the events read back from this module. If creation of the buffer was successful, a begin of run event (BORE) is written into the buffer, containing module parameters and configuration information later used in data analysis. This event also indicates the WTR that the run started up correctly.
If all modules have been initialized successfully, a wakeup message is sent to the writer causing him to call its start of run function and prepare everything for data taking. Afterwards, the BB IRQ is enabled, so BB IRQs will alert the IT and the TLU trigger is enabled, and now the run runs.
Shutting down a run is done vice versa. First, the TLU trigger is disabled. After a certain time interval long enough to let the IT process BB IRQs still pending, the BB IRQs are disabled. Then, for each module included in the DAQ readout chain a shutdown function is called, which e.g. disables the module IRQs, a EORE (end of run event) is written into the buffer and the buffer is destroyed. The EORE will signal the WTR and the following processes that the run is now finished, so data files can be closed and histograms can be saved.
As the PDC does not keep track of the total number of events, one of the modules is chosen during startup, whose events will be counted. This number is compared with the DAQ goal every time an IRQ is fully processed to find out when the run is finished. So, the resulting number of events taken may slightly exceed the specified run goal, but I think this does not really matter.

11.2.5 The interrupt thread (IT)

The interrupt thread, or, to be more precise, the instance of `TProcessInterrupt` is the process which is the genuine producer, the DAQ thread. It is a member of `TInterfacePCI` and is generated together with an instance of this class, so each BB PCI interface present in the system has its own IT. The IT is what is alerted, when an IRQ from the BB PCI interface is detected by the OS. A message from the interrupt service routine makes the `execute` function of the IT enter the IRQ processing loop. First, the source of the BB IRQ (timeout, XInt error, module) is detected, and after detection the appropriate handling function is called. For the module IRQ this is the `HandleModuleInterrupt` function. After handling the IRQ, the BB IRQ bit is cleared and the IT returns into the idle state until the next BB IRQ occurs. The `HandleModuleInterrupt` function is the genuine DAQ function. DAQ speed and mode strongly depend on how this function looks like. So far, two DAQ modes have been implemented:

- The autonomous IRQ DAQ
- The single event DAQ.

In the following, these modes will briefly described.

Autonomous IRQ The autonomous IRQ DAQ mode is the normal case and the case of the choice if only BAT modules and other BB modules capable of IRQ generation and data buffering are connected to the system. Each module takes data independent from the other modules. Triggers are coordinated by the TLU, but this is the only connection between the modules. Each module generates IRQs autonomously whenever the module's individual internal IRQC is met. The data is read back as soon as the IT has time to process this modules IRQ. Normally, this is right after the IT detects this module having caused the IRQ.
The `HandleModuleInterrupt` function loops through all connected modules, reading back all modules' ISR to detect the module that caused the IRQ. Then, this module's `InterruptAction` function is executed. The IT, being a member of `TInterfacePCI`, has access to all BB module instances by using the pointers stored in the receiver list administered by the current instance of `TInterfacePCI`. `InterruptAction` is a virtual function, defined in `TReceiver`, but implemented in the particular child classes, so the appropriate `InterruptAction` function is called for each connected module. Normally, the `InterruptAction` function accesses the BB module's FIFO,

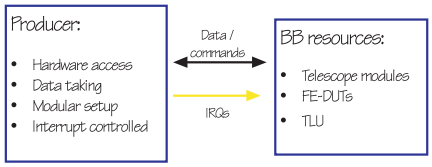


Figure 62: Autonomous IRQ DAQ structure looks quite simple. It only works if only BB modules are connected to the system. The IT receives the IRQs and reads data out of the different modules. Each module acts independently.

reads all data connected, separates it into data packages containing the data belonging to one event and sends it into the modules shared buffer as a series of module events. After finishing this, the module and BB IRQ are cleared and the system waits for the next IRQ.

Important is here, that data taking continues while the module's IRQ is pending[106]. So only in this DAQ mode, the high speed capabilities of the BAT can be fully exploited. Of course, the IRQCs for the different modules can be different. The system can run in autonomous IRQ DAQ mode also if one or more modules are in EIRQ mode, but due to the special properties of the EIRQ, full speed operation can only be realized, if all modules run in DIRQ mode. Autonomous IRQ DAQ mode means, there is no fixed order in processing the IRQs.

Single event DAQ The single event DAQ was implemented for the ATLAS testbeam periods at H8 testbeam at CERN in 2001, when VME based DUTs were to be read out which were capable of neither interrupt generation nor data buffering. All BB modules are operated in EIRQ mode. Every time an event occurs, an IRQ is generated. Without data buffering, each module has to be read out every time. A fixed order was introduced, in which the IT loops through all connected modules and reads their data, clears the IRQs and sends it into their shared buffers:

1. VME DUTs are read first,
2. the VME TDC is read next and
3. the modules are read in the order corresponding to their BB address.

[106] As explained in the section about the MFPGA, the module stops data taking only if the IRQ is not processed for a very long time due to prevent an impeding module FIFO overflow.

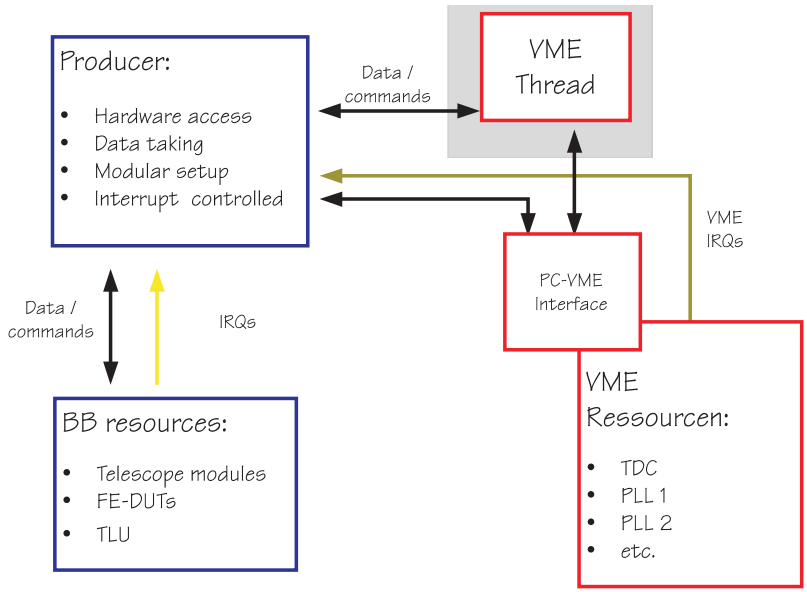


Figure 63: Single event DAQ structure. A fixed order of processing has been introduced. BB modules are read out last. The VME thread is used to process VME modules which are capable of data buffering (TDC). The VME thread polls on the fill level flag of this modules buffer to find out when to read it out.

This order was chosen, because the VME modules didn't give a busy signal. So, due to the properties of the EIRQ for the BB modules, which blocks triggers until the module's event has been read back, the BB modules had to be read last to maintain trigger synchronity. Only event rates of 1-2 kHz have been achieved with this DAQ mode due to the dead time being not only sequencer runtime but sequencer runtime plus readout time for all modules. Fig. 63 shows the schematic connection. The access to VME is done with a commercial PC-VME interface. For this DAQ mode, there is room for improvements. The BB modules, for example, could work in buffered mode if an IRQ would be generated by the TLU. In this case, the dead time would not include the readout time for the BB modules for every trigger. This has to be fixed as soon as possible.

11.3 Shared buffers

Shared buffers have proven to be a very useful and versatile tool for inter-process communications. They can be used to send large amounts of data from one process to another with high speed and comfort. They were developed by Peter Fischer in March 2001 based on windows API functions.

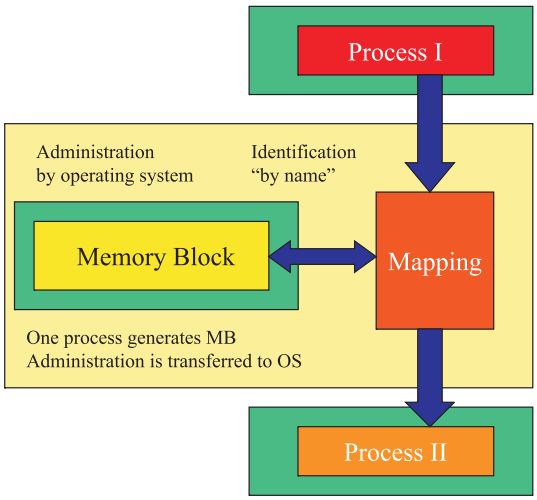


Figure 64: Shared buffer concept.

11.3.1 Concept

In BAT DAQ, shared buffers are used as one-way data highway to send data as well from PDC to WTR as from WTR to the OMO processes. They consist of a memory block in system RAM, whose size can be selected in the constructor and which is administered by the OS, and a couple of member functions acting as an interface to access the memory block from the application. To create a new shared buffer, the size and a buffer name has to be specified. The buffers are identified in the OS by their name. If another application tries to create a buffer with the same name, this buffer will not be created, but this application will gain access to the same memory block previously created by the first application[107] due to a process called *memory mapping*. Using windows API objects (so-called *mutexes*) it has been made sure that there will never be an access conflict when different processes try to access the same buffer. In figure 64 the principle of a shared buffer is shown.

The information stored in the Buffer is organized as array of integers. Amounts of data are counted as numbers of integers. The size of the buffer has to be given in number of integers. Access to the buffers contents is gained using integer pointers. If one wants to write a certain amount of data into a shared buffer, the function `GetWriteAddress` has to be called, with an argument indicating how much integers are going to be written into the buffer. The return value is an (integer type) write pointer, which can be used to write the data to the buffer in the same way like writing in an array. If the

[107] In this case, the buffer keeps the size specified by the first application. The size given by the second application is ignored.

space left in the buffer is not sufficient for the amount of data specified in the argument, the returned pointer will be `NULL`. When writing into the buffer, please note that the pointer has to be incremented any time an integer is written into the buffer. After finishing writing into the buffer, one has to call the `FinishWrite` function, passing the incremented pointer pointing to the next free space in the buffer as an argument. The buffer then knows, that, and how many, space was occupied. Please note that you can write less data than the value requested when calling `GetWriteAddress`, but never more. This will result in an access violation.
Reading from the buffer is very similar. Calling the function `GetReadAddress` returns a pointer, from which buffer data can be read. The function demands a second integer pointer as an argument. Accessing the variable this second pointer is pointing to after calling `GetReadAddress` yields the amount of data which is contained in the shared buffer and also can be read. After reading a certain amount of data, calling `FinishRead` with the pointer to the first unread buffer cell makes the buffer mark the space read as not occupied, so it can be overwritten. You can read less data than contained in the buffer (remaining data will stay in the buffer until it is read), but you should not read more (this will yield rubbish or access violations). Please note that the data read always remains in the buffer until the `FinishRead` function is called, so, different from a FIFO structure, random access to the values is possible.
Additional functions allow an overview about the buffer's state (full, empty, fill level). A buffer shared by two or more applications will stay active in memory until the last application sharing the buffer is terminated or actively destroyed the buffer.

11.3.2 Events

The data which has to be transferred from one process to another in BAT DAQ consists from events. An event is simply the data generated by a module connected to the DAQ readout chain for one trigger. To transfer this data through the shared buffer, to separate the events from one another and to make the events individually accessible in the shared buffer, a *buffer event structure* has been implemented. A buffer event consists of a data bulk and a event header. The event header consists of two words. The first word is an identification word containing in its 16 MSBs an identification bit pattern telling the user which kind of event it is and in its 16 LSBs the event size, the number of integers the event consists of, including the buffer event header. The second word is a trigger number word, which has especially been

implemented for telescope use to ease the task of trigger number verification by the WTR. Using this second event header word, the WTR can "at the first glance" check, if the trigger numbers from the first events in all buffers have corresponding trigger numbers.
What the producer does while reading the data from the FIFO is copying the data from the FIFO directly to the shared buffer leaving space for the buffer event header in front of the data bulk[108] until end of event data is read out of the FIFO (see the section about the MFPGA implementation). Then, the appropriate header is calculated, determining the final size and using the trigger number information received in the end of event data. Then, the event is complete. With a `FinishWrite` it would be receivable for the writer, but for efficiency reasons, a `FinishWrite` is done if all events contained in the FIFO have written into the buffer, each being equipped with its individual header.
Using this buffer event structure, easy access, readback and copying of data from one buffer to another or to hard disk, is possible. An event structure like this is always useful, when data is to be transferred which comes in packages of well defined size.

11.4 The Writer (WTR)

The Writer process does not only write events on disk. The WTR does also what is called *event building*; it collects the events coming from all the different modules connected to the DAQ readout chain, the so called *module events*, belonging to the same event (as indicated by their trigger number) and fills them into a structure called *BAT event*. What is stored to disk is a series of BAT events.
At run startup, the WTR receives a message from the PDC indicating that a run is about to be started. The PDC does this after having initialized all modules included in the readout chain, after having created a shared module buffer for each module and after having written a BORE for each module. The WTR starts to seek for module buffers after receiving the PDC message. The WRT configures itself depending on which type and how many module buffers it is able to find. When the initialization is complete, the WTR creates the OMO buffers if demanded, checks if there is a BORE for every module buffer[109], writes the BOREs into the OMO buffers and send the wakeup message to the OMO processes.

[108] This is possible only for the shared buffer with random access capability. It would be impossible for a FIFO structure.
[109] By convention, BOREs have trigger number 0.

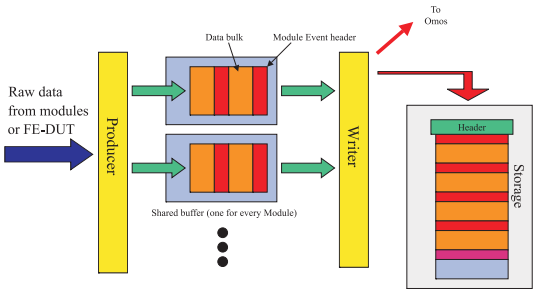


Figure 65: Structure of stored data.

The WTR does not necessarily write a BAT data file to disk. If it is a test run where the DAQ readout chain is to be tested or if the WTR only serves as interstation for the OMO data, the WTR can be configured in a way that a data file is not written. But if a data file is to be written, the WTR chooses the data file name absolutely and totally incremental, depending on prior run data file names. The data file is opened immediately after startup. Of course, the BORE is written to the data file too. The WTR is timer controlled. If all shared buffers are empty, the WTR waits for a certain time interval to pass before it looks into the shared buffers once again. The WTR becomes active, if all module buffers contain data. This is, because the WTR only writes full BAT events to disk, and therefore each module has to have sent at least one event forcing the associated shared module buffer to be not empty[110]. The WTR then looks into every buffer, compares the trigger numbers of the different module events. If they are different, something went terribly wrong and the run aborts. Otherwise, the WTR will write a BAT event header to disk and simply copy the module events, containing module event header and all, just behind the BAT event header, in a certain order. Details on this issue can be found in the documentation about the BAT data output format. The structure of such a BAT event is shown in fig. 65. Of course, the data size contained in the BAT event header is the overall data size, containing all module events and the BAT event header. The WTR can build and sent special events to the OMO processes. Which kind and how many, depends on the writer configuration. So far, three writer configurations have been implemented:

- BAT OMO only. No data is sent to a DUT OMO, there is not even a DUT OMO buffer opened. Only the BATDAQ BAT OMO is connected to the system. This operation mode is for telescope-only operation. A

[110]This is, why in the BAT environment modules who have no data do not send no data but a message containing th information that there was no data for a given event.

BAT OMO event is built, containing the module events of the BAT modules connected to the system.

- BAT and DUT. Two different OMO processes exist, one decoding and displaying the telescope data and one processing the DUT data. This mode was used for 2001 testbeam operation of the BAT. The existing DUT OMO is capable of processing ATLAS pixel front end data of any kind. A BAT OMO event, containing all BAT module events, and a DUT OMO event, containing all DUT module events, are built and sent into their associated buffers.

- All in one. The same event as is written to disk is sent into a shared buffer. This mode was not yet used so far, because it requires a common OMO process, capable of processing all kinds of data. Although such a process would be useful, it has not been developed so far.

The fraction of events sent to the OMO processes can be selected. This is, because the OMO processes are very CPU time intensive due to a lot of display stuff, forcing the system to slow down if they are to process to many events. A fraction of every tenth event has proven to help the online user to see what's going on, to provide the offline user enough statistics to have OMO histograms comparable to the offline results and to burden the CPU not more than necessary with graphics stuff. The writer stops processing data if either a trigger number error occurs or if an EORE is found in every module buffer. A trigger number error indicates that for some reason the triggers got out of track, either because of a module not processing a trigger (an exception which is to be caught by the TLU busy check) or because of an event getting lost somewhere between BAT module and writer. Both cases are even bad. An EORE is found if the trigger numbers for all events in all module buffers is 4294967295, being 0xFFFFFFFF in hexadecimal representation (an event number which would correspond to a REALLY large run), independent from the previous trigger number.

11.5 The OMO processes

In contrast to PDC and WTR, the OMO processes are not unconditionally necessary. WTR and PDC can without problem take data on their own. The OMO processes can not perform a detailed analysis of the acquired data. They are supposed to enable the user to supervise the system's performance by looking at a reasonable fraction of data acquired by the data acquisition chain. To this data, some simple signal processing algorithms are carried out

and the result is displayed in a suitable format, mostly histograms which are displayed graphically and updated continuously on line. The OMO task is to detect errors and mistakes not detected by the preceding data acquisition and analysis steps, or, where this is not possible, to enable the user to detect the errors by comparing the histograms generated with the on line data with some reference histograms displaying should-be data. The first case of errors occurs due to data transmission problems. An OMO task will alert a user if it can not decode a data package correctly. The latter case occurs if e.g. due to some configuration or operation error a module produces weird data in a correct format.
As described, the writer sends a selectable fraction of events in a specified format in a shared buffer structure dedicated to a certain OMO process. Data is written in this OMO buffer event by event. At the beginning of a run, the WTR sends a message to all OMO processes connected causing these processes to start up. The WTR does this right after it created the shared buffers serving as connection between writer and OMO processes and writing the appropriate BOREs into them. After starting up, the OMO processes expect to find the shared buffer and to receive the BORE. The BORE is sent in any case, independent from the selected event fraction. The BORE then tells the OMO process, which, and how many, modules are connected to the DAQ chain. With this information, the OMO process can configure properly, generating the correct number of events, arrays of the correct size, the correct number of histograms and so on. The OMO processes read the data event by event from the shared buffer. For each event, the OMO updates all its histograms. As the graphical output is what takes the most CPU time for the OMO processes, the rate at which the graphical output is refreshed can be selected. As soon as a run is finished, the histograms generated by the OMO process are stored, if wanted, in a subdirectory of the storage location of the current run data file. On demand, a different storage location can be chosen. These OMO histograms serve as reference data for further data analysis.
So far, two different OMO processes have been designed. The BAT OMO is suitable for decoding the BAT data of an arbitrary number of BAT modules only. A screenshot of this OMO program is shown in figure 66. The DUT OMO was designed to decode and display data from the DUT types the system was operated with so far. These are:

- ATLAS FE-DUT single chips
- PLL based ATLAS FE single chip devices
- PLL based ATLAS FE modules

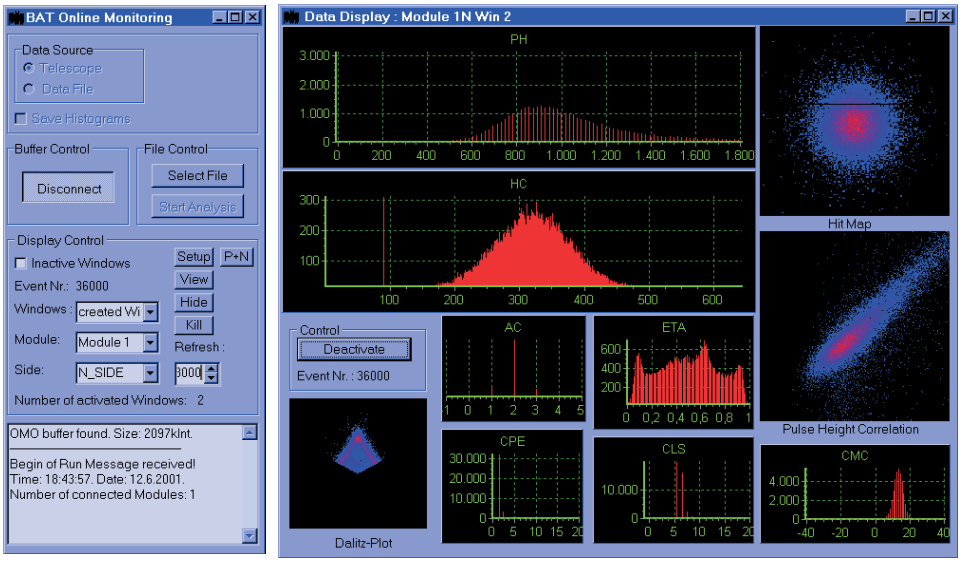


Figure 66: Screenshot of the BATDAQ BAT OMO process. The small left window is the control panel, from which several windows like the right one can be generated. The right window displays the data of one side of one module and two plots in correlation with that module's other side. The displayed histograms are: (left) Pulse height distribution, hit count histogram, Dalitz plot, (right) hit map, pulse height correlation (both in correlation with the module's second side), (bottom) η distribution, number of clusters per event, number of channels per cluster, average cluster and common mode distribution. In the hit count histogram, a noisy strip can be seen, which causes the excess in the middle of the η distribution. The strange shape of the η distribution is caused by the low energy of the ^{90}Sr- β source used for this test measurement.

- a CAEN TDC

BAT OMO and DUT OMO were both used in the first testbeam periods the BATDAQ was used in and brought satisfying performance. A common OMO process capable of decoding and displaying both DUT and BAT data would be useful, for then it would be possible to get an online view of correlations between telescope and DUT data. Such a tool, the WTR is already suitable for, is currently under development.
A stand-alone DUT OMO is the only program which has to be written especially for a new DUT. The WTR won't have to be modified at all, and into the PDC readout chain only additional classes will have to be included in a well-defined way.

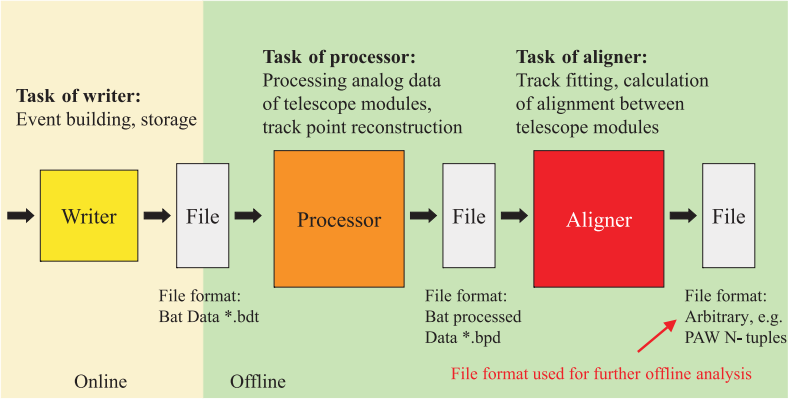


Figure 67: Flow chart of general purpose offline analysis. The BAT data files written by the writer are processed by teh processor, which performs a position reconstruction algorithm. The Aligner transfers the hit coordinates obtained in this way into a common coordinate system. The output data file of this task, which is still to be specified, will serve as input file for subsequent analysis task taking DUT data into account, for as well Processor as Aligner only look at telescope data.

11.6 Further processing

BATDAQ is considered to be a software package for testbeam data acquisition. At this point, this task is finished. For a more detailed analysis, some other, more sophisticated and CPU time intensive tasks will be necessary. Figure 67 shows a flow chart, displaying how data analysis based on the data files acquired with BATDAQ may look like. This offline package is still under development.

The task of the offline analysis is to provide the user a general purpose analysis tool especially for the telescope data. The DUT data will pass the entire process without being touched, and the BAT offline analysis will work independent from the kind of DUT connected. In a first step, the data files written by the writer are analyzed by a task named *Processor*. The Processor decodes the telescope data and performs a position recognition using the η distribution or a similar procedure to evaluate the analog data for better spatial resolution. Hit ambiguities are solved, and where this is not possible, the events are discarded. The output file for the Processor, a "BAT processed data" file, has reduced the analog cluster information of each telescope module to a two dimensional hit coordinate with reference to the module's origin (the upper left detector corner when the module is oriented like in fig. 15). The following task, the *Aligner* transfers the hit coordinates for each module into a common coordinate system. Or, in other words, the hit coordinates

will be given after the aligning process with respect to the POR mentioned in the introduction. For convenience, the POR is chosen to be the detector plane of module one. The transfer of the other hit coordinates into this common coordinate system is done by applying a transformation

$$\vec{x}_i^{\,com} = \mathcal{A}_i \cdot \vec{x}_i + \vec{\mathcal{V}}_i$$

on each hit coordinate, with $\vec{x}_i$ being the initial hit coordinate for module i, $\vec{x}_i^{\,com}$ the hit coordinate with reference to the POR, $\vec{\mathcal{V}}_i$ a vector describing a translation and $\mathcal{A}_i$ a vector describing a rotation. Those alignment parameters are different for every module and have to be obtained in an iterative χ^2 minimization process which is the major task of the aligner. Once the alignment parameters have been calculated, everything is easy.
The last thing that remains to be done for the Aligner is the track fitting. Using the hit coordinates with reference to the POR and their errors, one can easily calculate a track by determining angle and position of the track with reference to the POR[111]. The output data file of the Aligner, whose format is not yet specified, will contain exactly this information: The track parameters with reference to the POR and the untouched DUT data for further processing in DUT specific offline analysis processes. What remains to be done there is calculating the alignment of the DUT with reference to the POR and, of course, decoding of the DUT data. And then, the real analysis work can begin...

[111]This is valid, of course, only if a straight line fit is sufficient to describe the track. In the presence of a magnetic field, a more complicated track model will have to be applied.

Figure 68: The BAT inside the H8 testbeam area at CERN. The four modules are fixed to a solid support each mounted on a stone table. The H8 testbeam magnet can be recognized in the background. A DUT (ATLAS FE single chip) not yet fully wired is located between the first and the second BAT nodule. The trigger scintillators are also visible.

12 Outlook

At this point, the description of the telescope system at the present status is complete. The telescope hardware and the BATDAQ software package have successfully been operated at the H8 testbeam facility at CERN. Figure 68 shows the BAT in the H8 testbeam environment.

Of course, there is still a lot to do. There are several things to be improved and a lot of features offered by the system are not fully exploited. Nevertheless, the BAT is already at the present state a flexible and useful instrument for testbeam data acquisition. Moreover, it it possible to develop the system even further to keep track of recent developments in the field. Some of the further developments are described below.

- *Replacement of detectors.* Due to the modularity of the system, detectors can easily be replaced by simply replacing the hybrid assembly,

if e.g. because of long-term operation a detector's noise performance worsened. The prototype system was built for double sided strip detectors of 3.2×3.2 cm^2 size, but with slight modifications to the case operation of a lot of different detector arrangements is possible. In applications radiation length is not that important as it is at the ELSA testbeam facility, one could operate two single sided detectors with a stereo angle of 90°. Also, detectors or hybrids of a different (smaller) size can easily be inserted into the system[112]: the modifications to the mechanics are easy and the changes to the MFPGA programming are well-defined. Furthermore, one can think of different detector types, if wanted, e.g. single or double sided strip detectors with one or more interstrips. In this case, the system's software will have to be changed only marginally.

- *Replacement of analog electronics.* If the changes to be made to the system are more radical, modifications of the analog electronics may be necessary, too. By keeping the DTC one can think of operating a telescope with pixel detectors as sensitive elements by changing the MFPGA sequencer programming, if only the appropriate analog electronic can be provided. The software changes, which in this case will be unavoidably necessary, can be made similar to the case of integrating a BB based DUT. There, use is made of the software's class structure and its well-defined BB interface, which is capsuled in the parent class of all BB modules.

- Even if the DTC hardware's resources prove to be insufficient for the demands which arise from a modified concept, the system's core can be kept as long as the BB communications resources are not touched. One can think of replacing the MFPGA with a larger type, maybe from a different family (VIRTEX etc.) that might make even the external RAM and FIFO unnecessary. This is surely worth while, as the progress in the FPGA field made in the last 3 years since the development of the DTC comes together with the reliable, fast and well tested BB system. But if speed and capabilities offered by the BB turn out to be insufficient also, maybe a faster and more reliable bus system is available for the same purpose, and resources exist to adapt hardware and software. This would be the only case in which a redesign of the system would be more favorable than an attempt to adopt the existing concept.

[112] This is valid as long as the hybrid layout and concept stays the same, for hybrids bearing a larger number of VA ICs additional bias voltages and currents may be needed.

Keeping this in mind, one can see that the BAT is more than just a test-beam telescope, but can, in its core, serve as a platform for a lot of different reference detector and data acquisition concepts, thus permitting a constant advancement on the field of reference data acquisition for position sensitive radiation detectors.