

ARCS
APV Readout Controller Software

**A Graphical User Interface
for Hybrid and Module Testing**

Version 5.01

by

Markus Axer, Franz Beißel, Clemens Camps,
Torsten Franke, Can Engin Ilgin, Stefan Kasselmann,
Jan Niehusmann, Andreas Nowack, Michael Pöttgens

III. Physikalisches Institut B
RWTH Aachen
7th August 2002

Contents

1	Introduction	2
2	Initialization Phase	4
2.1	Configuration Files	4
2.2	Initialization	5
3	Monitoring and Controlling Phase	8
3.1	The Index Cards of the Monitoring Supervision	9
3.1.1	APV-Display	9
3.1.2	APV & Boards	10
3.1.3	Errors	11
3.1.4	MUX & PLL	11
3.1.5	Data	12
3.2	The Hybrid Current and Voltage Supervision	12
3.3	DCU Supervision	13
3.4	The Data Panel	13
4	Automated Fast Tests	15
4.1	Invocation	16
4.2	Running the Tests	16
5	Deep Tests	18
5.1	Pedestal, Noise and Common Mode Calculation	19
5.2	Pipeline	20
5.3	Pulse Shape Computation	23
5.4	LED Test	25
6	All Tests	28
7	TCP/UDP Connection	30
7.1	TCP Connection	31

7.1.1	Sending nominal temperature via TCP	31
7.2	Receiving Data via UDP	31
7.3	Data Format	31
7.3.1	UDP Data Format	32
7.3.2	TCP Data Format	32
8	VIs	33
8.1	APV_Register	33
8.2	Trigger	35
8.3	DAC Offsets	36
8.4	PLL Control	36
8.5	LED Controller	37
8.6	DCU Test	39
8.7	Analysing ARCS data with LabVIEW2Root	40

Chapter 1

Introduction

The **APV Readout Controller Software (ARCS) Version 5.01** is a LabVIEW 6i application that serves as graphical user interface in the module and hybrid test setup (ARC) developed in Aachen. This test setup is composed of:

- the **APV Readout Controller (ARC)** board (one board is suited to control two modules/hybrids),
- the **ARC FrontEnd Adapter (ARC FE)** to which a module/hybrid is connected directly,
- the **PCMIO** interface that fits into an IBM XT slot and serves as interface between the ISA bus and the ARC board,
- a **Power Cable** (3 banana to D-SUB 9, red banana: $+5V$, black banana: $0V$, blue banana: $-5V$),
- grey **ISA Bus cable**,
- coloured **twisted pair cable**,
- a standard Power Supply ($\pm 5V$; min. $0.5A$ on the $-5V$; for the ARC board $1.4A$ and $1.0A$ per hybrid on the $+5V$),
- the **ARC Software**.

The setup can be extended by:

- the LED controller board **LEP 16**,
- the HV module **DEPP**.

For details and pictures see <http://www.physik.rwth-aachen.de/group/IIIphys/CMS/tracker/en/index.html>

ARCS is a software application that is intended for **data acquisition (DAQ)** and test purposes. The communication between the setup hardware and the operator is based on functions written in C/C++ while the LabVIEW environment is used as graphical user interface. The C/C++ code is embedded in LabVIEW as libraries (**dynamic link libraries (DLL)** for WINDOWS systems or shared libraries for LINUX systems). ARCS is successfully tested and used with WIN9x so far¹.

The ARCS package is available on our web site: http://www.physik.rwth-aachen.de/group/IIIPhys/CMS/tracker/software/arcs_v501.zip

¹The portability of ARCS to WINDOWS NT/2000 was studied and successfully performed at Fermilab. We thank these people very much!

Chapter 2

Initialization Phase

2.1 Configuration Files

The ARCS package consists of one executable file and two configuration files. The configuration files are named *ARCS_main_config.cfg* and *APVdefault.set*.

ARCS_main_config.cfg is intended to define the used type of PC interface (most of the ARC users will use the ISA interface, so the corresponding entry has to be "1"). In addition it is possible to define the directory paths where data or test results have to be stored.

Note: If you change the name of this configuration file or if you change its location (the file has to be in the same directory where the application is executed), ARCS will not be able to read it! It is not mandatory to use this file: If you rename or delete the file, some default settings will be used for the rest of the readout process (this means: ISA interface, all created files will be written into the same directory where the application is executed).

A reasonable usage of *ARCS_main_config.cfg* requires the following file format:

```
*****Types of Interface: 1=PCISA, 2=EPP(0x278), 3=EPP(0x378)
Interface :           1                               ... must always be in line 2!
*****Directory for Data Storage:
C:\ARCS\Data           ... use your own directory; must always be in line 4!
***** Testcenter, Operator, Software Setup
Center:  ...
Operator:  ...
Version 5.01
```

APVdefault.set includes default register settings for APV25-S1 chips as specified in the APV manual. This file (if available) is used for the initialization of the APVs. So you can feel free to edit the file to initialize the APVs with special register values. The file format looks like this:

43 Mode
4 Latency
98 IPRE
52 IPCASC
34 IPSF
34 ISHA
34 ISSF
55 IPSP
34 IMUXIN
0 ISPARE
29 ICAL
30 VFP
60 VFS
40 VPSP
254 CDRV
1 CSEL
4 MUXGAIN
0 ERROR

2.2 Initialization

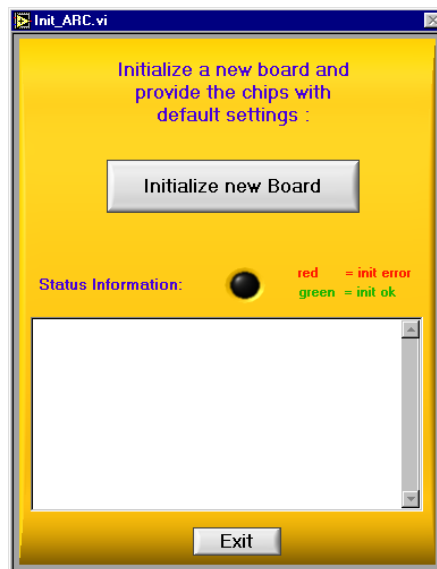


Figure 2.1: The Init_ARC.vi is the first step of the initialization procedure.

One of the most important and sensible steps within the data acquisition process represents the initialization phase. The process includes the initialization of the ISA bus, the I²C bus, the chips etc.. After starting the ARCS application, a dialog window appears: Use button

Initialize new Board to define a new board setup (see figure 2.1).

This opens the “ARCS_status” window where you can choose between the following options (see figure 2.2):

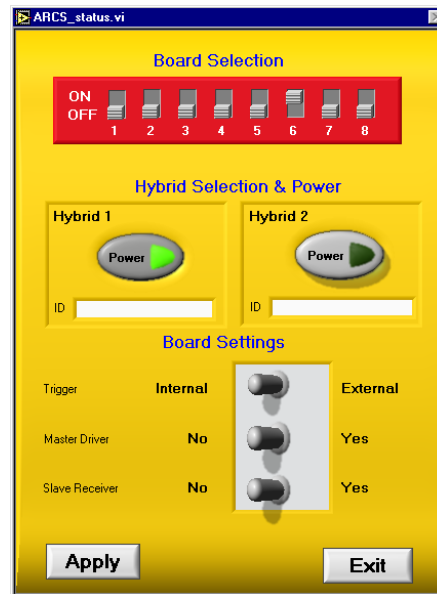


Figure 2.2: The ARCS_status.vi

- Board selection* (red) dill switch: the switch settings have to be the same as the similar (red or blue) switch on the ARC board you want to initialize; you’ll never see any kind of APV frame if the equality is not warranted!
- Hybrid selection and power* select hybrid 1 and/or 2 and turn the hybrid power on/off (have to be switched on to communicate with the APVs...). You may control the success of this transfer by observing the LEDs on the front side of the ARC board. Underneath the power control button, you can find a text field, in which you can fill in a Hybrid ID (barcode), either by a keyboard or a barcode scanner.
- Board settings* are intended for advanced purposes (in a multi board setup, using external trigger). You may choose the following features:
internal/external trigger (can also be done later, see section 8.2)
master driver: yes/no (can only be done here)
slave receiver: yes/no (can only be done here)

the default values are: *internal trigger, no master and no slave receiver*. Please leave the switches at their default settings as long as you run a *single board setup*.

Apply

loads all chosen settings. Don't push this button if you opened the initialization window accidentally, so no changes concerning the board setup will take place.

Exit

leaves the current window.

If the definition of a new board proceeded successfully, the initialization of the board and all chips takes place and will be finished with a (hopefully) successful I²C scan. Some error messages might occur during the initialization, providing also some possible solutions. The results of the initialisation phase are monitored in a message box and are summarised by a red or green error flag light. In case of a red light you can retry to initialize the board by repeating the previous steps.

If your set up consists of more than one board you simply have to repeat the initialization procedure for every additional board. After all of your boards are initialized (most of the people are using just one board) you can leave this part of the application by pressing **Exit**

Chapter 3

Monitoring and Controlling Phase

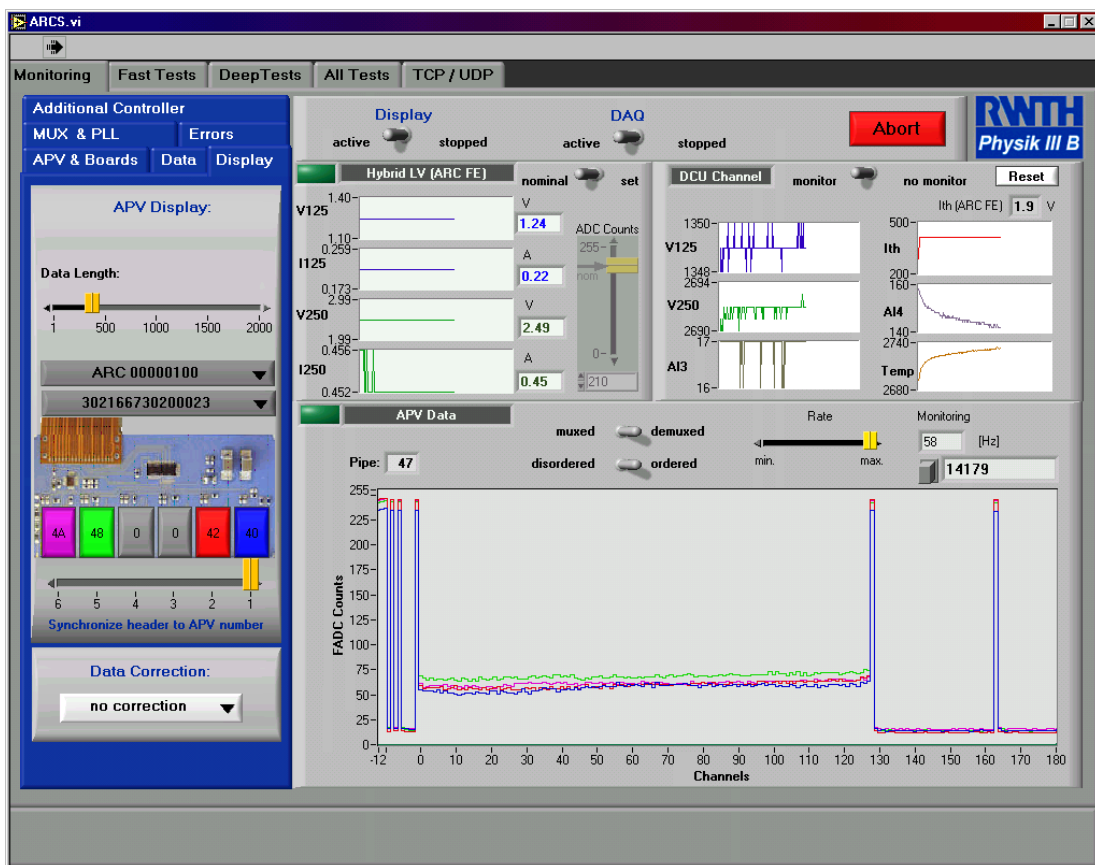


Figure 3.1: Screenshot of a typical ARCS display during the monitoring phase.

The ARCS software is subdivided into four main parts (*index cards*), that can be selected by choosing one of the following four buttons on the upper left margin of the ARCS window:

Monitoring	This is a kind of expert mode. All settings have to be done by hand. There are no automated functions. Descriptions concerning that part can be found in this chapter.
Fast Tests	This part consists of some basic tests that are useful to check the reliable operation of the ARC hardware and the I ² C access to the ASICs. For more details see chapter 4.
Deep Tests	This is a tool for online calculation of pedestal and RMS noise, for pipeline checks, pulse shape scans and LED tests. For more details see chapter 5.
TCP Connection	Here you can establish a TCP connection to any application (e.g. Aachen Cooling Device), that uses the defined TCP port. It is possible to get data from this device, if a special data format is used. For mor details see chapter 7.

3.1 The Index Cards of the Monitoring Supervision

Figure 3.1 shows the screen that appears once the monitoring phase of ARCS has been started. The main part of the screen is taken by the data panel on which the data read out is be visible. A number of index cards are situated on the left side, while in the upper right corner the hybrid currents, hybrid voltages, and the DCU supervision can be seen (only the six used channels are displayed).

3.1.1 APV-Display

This index card offers some options to influence the optical appearance of the data on the data panel:

<i>Data Length</i>	With this slide control the length of the data buffer, read from the ARC board, can be determined. The effect of an adjustment can be observed on the display panel (you are not able to see the whole data length if you are synchronized to a running APV as the pre header data will be cut off). Be aware that choosing a too short buffer length runs into problems with DAQ and calculation routines. You must be able to see a tickmark at least at “channel” 175 to have a reasonable DAQ. A long data length slows down the sampling rate, but it is useful if you want to send more than one trigger or run the APVs in the 3 sample mode (gives three frames in peak mode).
--------------------	--

ARC 00000100 ▼

The number after *ARC* represents an 8 bit pattern that correspond to the selected boards (dill switch) you previously initialized (see table in section 2.1 on page 6). The data display will be toggled to the data belonging to the chosen *ARC* board and hybrid (see below).

Hybrid_1/2 ▼

This button is used to toggle the data display between hybrid 1 and hybrid 2. Additionally the ID of the module respectively the hybrid ID is displayed after the hybrid number.

APV Display

shows a picture of the hybrid and offers the possibility to select the APVs that should be displayed on the display panel; to choose a special APV, just click on the position of the APV in the picture; chosen APVs are lighted up on the hybrid picture. Their data appears on the display panel in the same color.

synchronize header to APV number

Determines which of the APV data sets is used to find the header (a header is found when a tickmark is followed by a digital one with less than 30 counts distance). All other APV data will be output synchronized to the chosen APV.

Data Correction

Leaving the choice button at **no correction ▼** displays the data as it is output from the ADCs; **reset pedestals ▼** is used as preparation for taking new pedestal data. All necessary values within the pedestal calculation routine will then be set back to zero; while the choice button remains at **take pedestal data ▼** data for pedestal calculation is taken. **correct data online ▼** applies a simple data correction, i.e. the difference between baseline and data will be subtracted. So the data display will appear much smoother.

The choices “reset gain” and “take gain data” from ARCS 4.0 has now vanished since this calculation was very complicated and can now be done much easier using the deep test “pulse shape” (see section 5.3).

3.1.2 APV & Boards

Board Properties

DAC Offsets

opens a new window which allows to change the DAC offsets that are added to the analogue APV data; Values of BE(hex) resp. 190 (dec) are recommendable for convenient data (see section 8.3).

ADC Clock Adjust
[1.7 ns]

This dial is used to adjust the delay between the hybrid clock (APV, PLL etc.) and the board clock (clock for FADCs, RAM etc.). It can be set from 0 to 17ns in multiples of 1.7ns.

Hybrid Power Switches the hybrid power on or off. The hybrid is the same as the one that is displayed (see section 3.1.1).

APV Properties

Trigger	opens a new window which allows to set several internal or external trigger pattern or calibration requests. Some examples are also given (see section 8.2).
APV Register	opens a new window for setting the APV registers (see section 8.1).
RESET 101	sends a 101 reset sequence to all APV chips.
I2C Scan	repeats the I ² C scan and enters the found I ² C addresses in the choice box beside it.
Re-Assign	This button is foreseen to repeat the assignment procedure that is done during the initialization. Within this procedure one APV is switched to the 3 sample mode. Then the data is evaluated to see from which RAM we read samples with three frames.

3.1.3 Errors

Here you'll find some kind of status report of found errors, i.e. about the number of header errors (detected error bit in the header) and column errors (column numbers are not the same for all APVs). The error counting can be reset to zero using the **Reset Errors** button. (If this does not stop the errors try **101 Reset**, and then press **Reset Errors**.)

Additionally the software limits for hybrid current and voltage on the different lines can be set. In case of exceeding these limits a red lamp will light up on the hybrid current resp. voltage panel (see section 3.2).

3.1.4 MUX & PLL

MUX Control

Choosing several of the available switches connect the corresponding resistors in parallel to the data output line. So if high number of resistors are set, a low output signal is visible on the data panel whereas low values involve high signals on the data panel.

PLL Control

Pushing this button opens a new window, which allows to read and write the PLL registers CTR1 to CTR5. (see section 8.4 for more details.)

3.1.5 Data

The *Data* card is reserved for APV data storage and analysis purposes. The data will be stored in the directory chosen in the configuration file (see section 2.1). You can edit this file during the monitoring phase, so you are able to change the directory of data storage as often as you want (also during running ARCS). ARCS 5.01 provides the possibility to write down either raw data or preanalyzed data both in ASCII format.

Write LV, DCU Data opens a new VI window that offers a simple DCU check, see section 8.6 for further details. The data taking procedure can be interrupted by pressing button **Write LV, DCU Data** once again (the color of this button should change).

Write Raw Data writes all data displayed in the monitoring window to a file called “Raw_Data_30216630200023.dat”, if you connect a module with barcode number 30216630200023 to hybrid 1 port for instance. You can determine the number of events written to that file via the textfield beside the button (20000 events are enough for all kinds of sophisticated analysis!).

Note: before taking raw data make sure that data length is set to a sufficient value 3.1.1. An insufficient value of data length cuts the APV frames which causes some strange effects during later analysis parts. The APV frames have to be synchronized to each other, because the raw data storage routine simply writes down the first 141 channels of each APV displayed on the APV supervision. So just the digital header, the 128 channel data and the first tick mark after the analogue data are recorded. The data taking procedure can be interrupted by pressing button **Write Raw Data** once again (the color of this button should change).

Start LabVIEW2Root pops up a window which allows to specify the directory containing your data files to be analysed. For further details see section 8.7.

3.2 The Hybrid Current and Voltage Supervision

In the upper middle of the graphical interface the *Hybrid LV (ARC FE)* field is placed where the hybrid voltages and currents measured by the ARC Frontend Adapter are monitored. The axis of ordinates is given in volt and amperes with a resolution of 10 mA on I125/I250 and 10 resp. 20 mV on the V125/V250 line while the axis of abscissas is given in arbitrary units (roughly time in seconds). The little switch at the right side enables/disables a voltage control by hand. If it is switched to *nominal LV*, the default value of 210 ADC counts is set. Otherwise the slide controller can be used to supply different voltages to the hybrid.

Note: Since the voltage on the V125 line is always half of the one on the V250, you can only drive both in parallel.

3.3 DCU Supervision

On the upper right side of the screen the DCU outputs are monitored. Due to the lack of exact information concerning the DCU outputs, the units are arbitrary at the moment respectively showing the digital data read from the several DCU registers. No DCU calibration is done.

3.4 The Data Panel

The data panel monitors the APV data you requested. After the initialization phase the well known APV frames should be displayed if everything went right. This monitor can be used to get a quick overview of effects produced by e.g. changes of APV or PLL registers. The axis of ordinates represents the range of the ADC counts (0 - 255) and within the APV frame (header, address and analogue data) the axis of abscissas can be interpreted as APV channel numbers.

If you want to stop the display to have a closer look at stand still APV data, you can switch the display knob from *active* to *stopped* (on the ledge above the Hybrid current supervision). This will stop the display and increase the sampling rate. So, this is the preferred setting while APV data storage. Another possibility is to stop DAQ which means that the display and the data flow has been stopped. Alternatively you can reduce the rate of the display by adjusting the horizontal pointer slide **Rate**. Decreasing the display rate increases the data acquisition rate. This feature is useful if you want to take data quickly while having a look at the data from time to time. The display rate is notified in the textfield at the right of the slide.

To toggle the data display from *ordered* to *disordered* a button on the ledge between hybrid supervision and data display is foreseen. The APVs do the data output not in the physical order of the channels; by choosing *ordered* the first data set will be displayed in the physical order of the channels. Further data sets (visible, if more than one trigger is sent, see section 3.1.2) are still displayed disordered.

The APVMUX chip multiplexes the output of two APV25 chips onto a single output line; by toggling from *demuxed* to *muxed* the data will be displayed demultiplexed resp. multiplexed.

At the upper left corner of the APV frame window the pipeline address of the displayed frame is shown.

Note: The address is always the address of the APV frame read out at last. If the data is not synchronized, this address might differ from the frames you chosed to be displayed. A value of 255 indicates, that the pipeline address is invalid or could not be decoded respectively.

If you want to change the range of one axis, mark the outermost value and type in the new limit. With *return* this new setting will be taken!

The last exit possibility is given by the **Abort** button which immediately stops the application.

Chapter 4

Automated Fast Tests

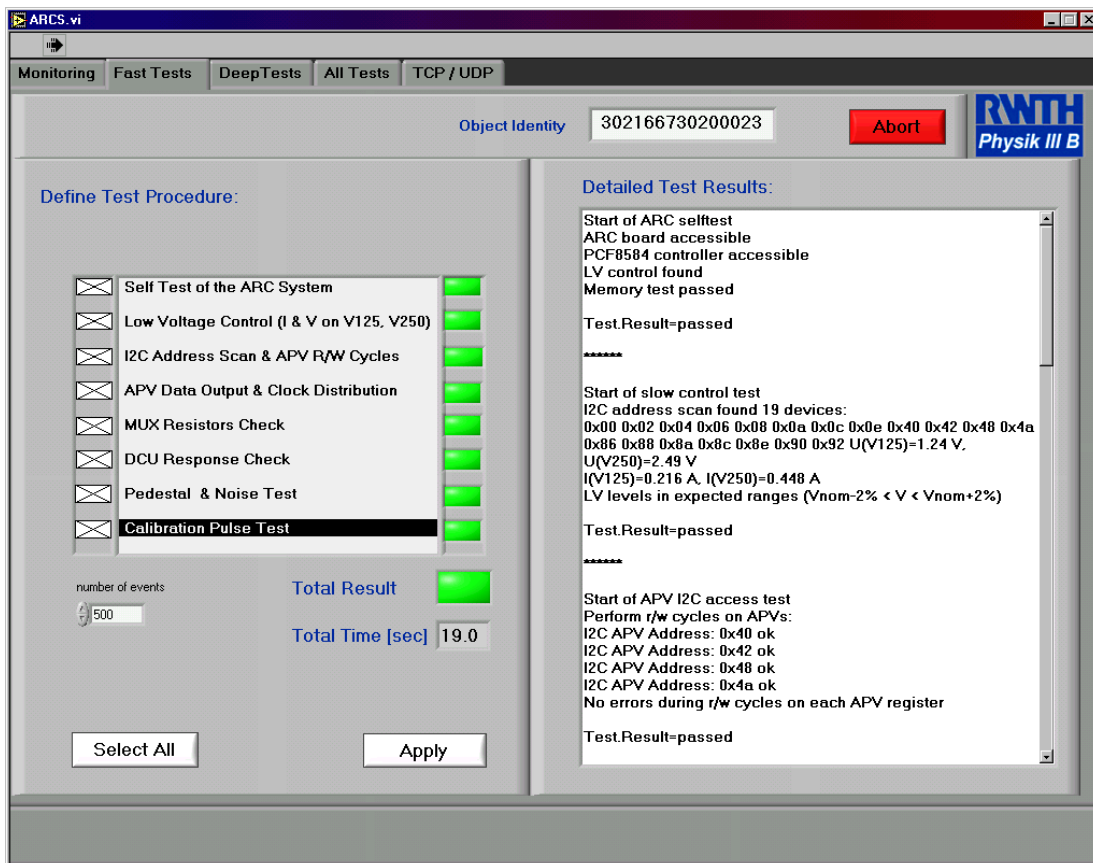


Figure 4.1: The user interface for automated tests.

The automated tests are meant to provide fast, standardized procedures to test basic functions of the hybrid. They can be operated without detailed knowledge of the hardware. These

tests are also used within the **Frontend Hybrid Industrial Test** procedures (FHIT).

4.1 Invocation

After starting ARCS as explained in chapter 2, the automated tests can be selected by the card index `Automated Fast Tests` on top of the screen. The text field *Object Identity* gives you information about the bar code of the currently tested object, if entered within the initialization procedure.

4.2 Running the Tests

On the left side of the screen, you can select which kind of tests you want to run (see table 4.1). Pressing the `Apply` button starts the tests. Furthermore you can choose the number of events you want to use for all calculations concerning data taking (default value: 100 events).

All results are saved to a log file which is named *ARCx_y.log* (with x= ARC ID and y=Bar Code) and which is written to the directory specified in the *ARC_main_config.cfg*.

Each test result is indicated with a red or green light. Additional information is displayed on the right side of the screen.

Self Test of the ARC System	checks the ARC hardware for errors: ARC board, PCF8584 controller access, I2C access to the LV control of the ARC front end adapter, memory test.
Low Voltage Control (I & V on V125,V250)	checks whether the voltages and currents on the 1.25 resp. 2.50 volt lines are within a range of $\pm 2\%$ of the default values.
I ² C Address Scan & APV R/W Cycles	scans for all I ² C addresses, compares the addresses to the address region of the particular chips, makes 10 R/W cycles to the APV registers.
MUX Resistors Check	switches on/off different MUX resistors and observes the changes of the height of the digital one and digital zero. The height should decrease with a higher number of switched on resistors. The decrement should be the same when different but equivalent number of resistors are switched on. The number of events for each calculation is given by the “number of events” you entered in the corresponding textfield.
DCU Response Check	this test varies the voltage supplied to the hybrid and compares it to the voltages measured by the DCU.
Pedestal & Noise Test	takes the number of events you entered and tests if the pedestals and noise vary in a reasonable range.
Calibration Pulse Test	uses the calibration logic of the APVs to generate signals of 2.0 mips. The signal is measured at a tight latency at the maximum of the shaped curve. The maxima have to be within a certain range to pass the test.

Table 4.1: Description of the automated fast test procedures

Chapter 5

Deep Tests

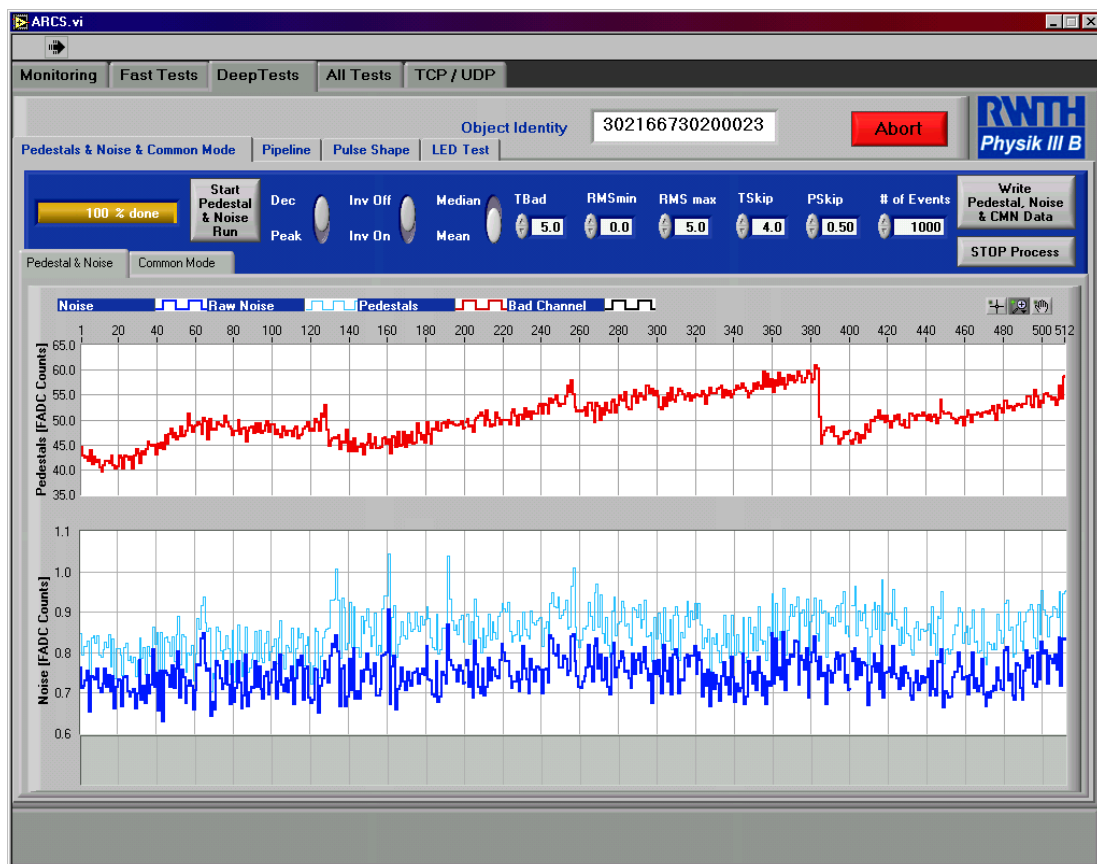




Figure 5.1: The user interface for the automated pedestal and noise test.

The deep tests (previously called additional tests) consists of four automated tests now: Pedestal, Noise and Common Mode Calculation, Pulse Shape Computation, APV Pipeline Check and a LED Test (requires the ARC LED Controller).

5.1 Pedestal, Noise and Common Mode Calculation

This test provides an automated online pedestal and noise calculation. The calculation routines are based on the “APV Analysis” classes by Valery Zhukov, which are also used for the long term tests (Thanks to him).

All APV settings you want to run the test with must be entered before, using the features of the monitor screen (see chapter 3). The calculation can be started via the  and can be stopped via the  button. A progress bar indicates the advances in calculating. The provisional results are continually updated while the test runs. The meaning of the different plots and colors is shown in the upper left corner of the panel. The upper plot display shows the pedestals (red lines), the plot display below shows the raw noise (thin light blue lines) and, as soon as a certain number of events is overstepped, the cmn noise (blue lines) is shown. The lowest plot display (light grey) is used to flag bad channels. The number of these flags should be stable when you run the test several times. The (dis)advantage of the new pedestal and noise calculation is, that it has many parameter to play with. The default settings in ARCS 5.01 seem to be convenient for module tests, but not for tests with hybrids (at least not for the one we could test so far). The meaning of each parameter is described below.

Within one event a channel is masked as a **noisy channel** in that event, if

- the signal $s(i)$ of that channel i is greater than $T_{skip} \cdot rms(i)$, where $rms(i)$ is the noise of that channel over all events; i.e.

$$|s(i)| > T_{skip} \cdot rms(i)$$

- or the signal $s(i)$ of that channel i is greater than the average signal times T_{skip} , i.e.

$$|s(i)| > \frac{\sum_i |s(i)| \cdot T_{skip}}{128}$$

For each channel the number of noisy events is memorized.

A channel is masked as a **bad channel** (and appears in the flag display) if

- the $rms(i)$ of the channel i deviates more than σ_{cut} times the rms of the rms from the average rms $\langle rms \rangle$ over all channels, i.e.

$$|rms(i) - \langle rms \rangle| > \sigma_{cut} \cdot rms_{ofrms}$$

- or the rms of that channel i is smaller than rms_{min} , i.e. $rms(i) < rms_{min}$, so rms_{min} represents the maximum noise that is tolerable before a channel is masked as dead.

- or the rms of that channel i is greater than rms_{max} , i.e. $rms(i) > rms_{max}$, so rms_{max} represents the maximum noise that is tolerable before a channel is masked as noisy.
- or that channel was masked as a noisy channel in more than $P_{skip} \cdot N_{rms}$ cases, where N_{rms} is the Number of events used for the noise calculation. So P_{skip} should be a number between 0 and 1.

All calculations can be done either by computing the mean or the median of pedestal and noise distribution. Use the correlative button to switch between both methods.

There are some additional parameters in the calculation algorithms, that are not accessible by the user. These parameters are:

N_{raw}	the number of events taken for pedestal calculation before the computation of the noise starts. It's value is always 20 percent of the # of Events.
N_{cmn}^{iter}	determines the number of iterations for the common mode substraction with flagged channels taken into account. It's value is always 1.
N_{strips}^{cmn}	determines the number of channels summarized as a group for the common mode computation in that group. It's value is always 32. You can also choose to do the tests in different APV modes

The files you create using Write Pedestal/
Noise Data consist of the barcode number of the module/hybrid, the mode (peak/dec) and the inverter setting (on/off), e.g.

PedNoi_30216630200012_PeakIon.dat

Additionally you can watch the common mode distribution (see figure 5.2) that was measured during the pedestal and noise run. The I²C address of each APV will be displayed in the upper right corner of each plot. If only four addresses were found during the initialization two of these plots will remain empty. Each diagram shows with thin lines the common mode distribution of each common mode group (32 channels) and the sum of all groups with thick lines, so the integral over all entries will give you the fourfold of the number of events. The colors in the plots correspond to the colors you know from the monitoring screen (see figure 3.1) The common mode can be measured in the range from -24.9 up to +24.9 ADC counts, with a bin width of 0.1 ADC count. A higher or lower common mode will be put in the overflow or underflow bins respectively.

5.2 Pipeline

The pipeline scan is a new feature of ARCS 5.01. It replaces the calculation routines for the APV_Matrix files in ARCS 1.0 - 4.0. The calculation is also based on the APV Analysis

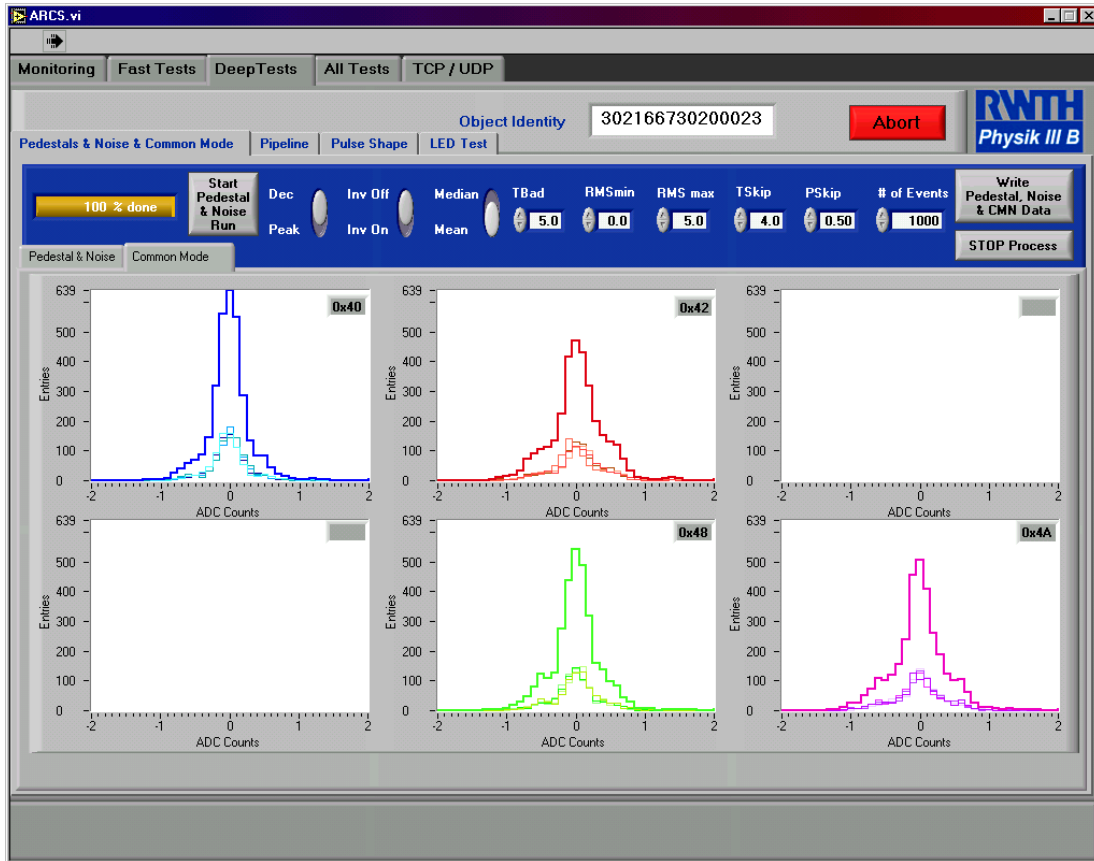


Figure 5.2: Display of the common mode distribution (with four APV on a hybrid).

classes by Valery Zhukov. It's goal is to find conspicuous cells in the APV Pipeline. The main difference to the previous calculation is, besides the new calculation algorithms, that the pipeline scan is no longer done with a random trigger.

Procedure: A tight value is written into the latency register of the APVs (value=10), a reset is send in the high byte (see section 8.2) of the trigger pattern, a single trigger in the low byte and the spacer is varied from 215 (to access pipecell 0/192) down to 24 (to access pipecell 1).

$$\underbrace{10100000}_{\text{highbyte}} \underbrace{215 \dots 24}_{\text{spacer}} \underbrace{00000001}_{\text{lowbyte}}$$

Since the pipeline scan uses the same computing algorithms as the Pedestal and Noise Calculation (see section 5.1) the parameters are of the same relevance.

During the measurement a progress bar shows the status while the number on the right of it shows the pipeline cell that was scanned last.

One can choose to take a look at the pedestal resp. noise matrix either from the channel or from the pipeline side. If you choose the look from the pedestal side, you can switch from

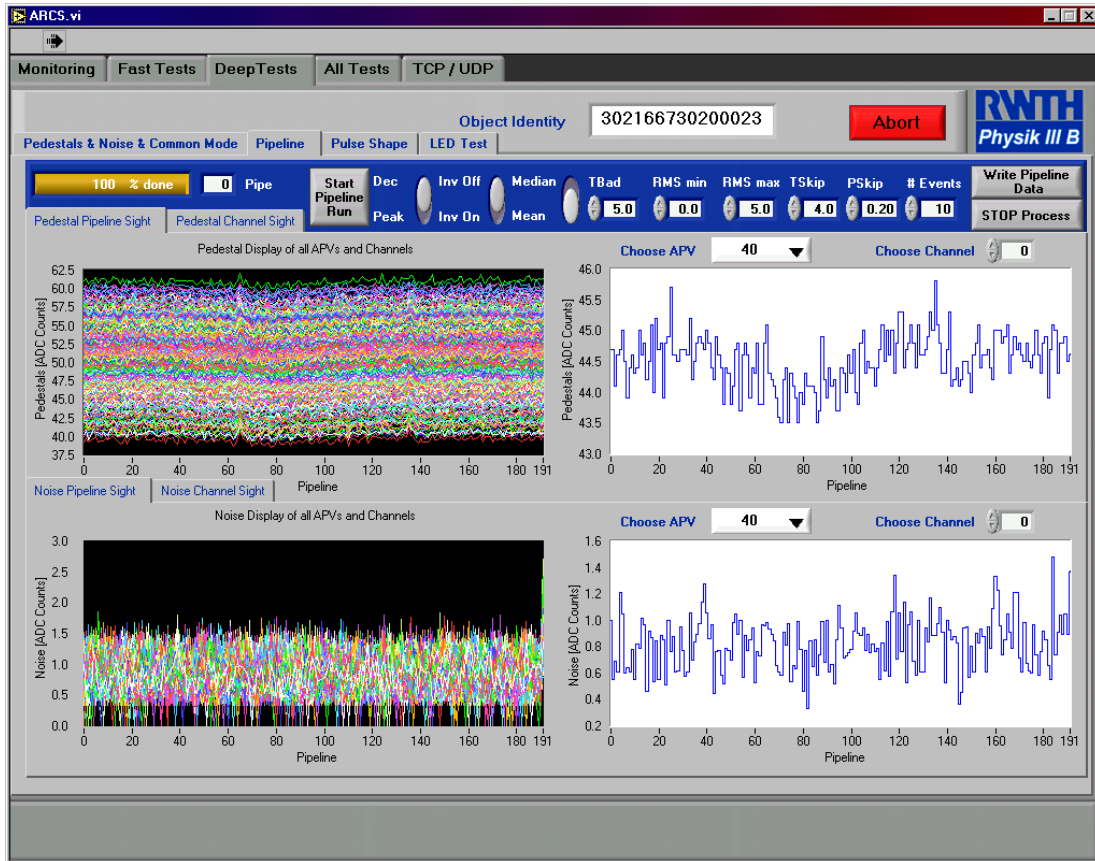


Figure 5.3: The user interface for the pipeline scan.

channel to channel, whereas, if you choose the look from the channel side, you can switch from pipeline cell to pipeline cell.

The files you create using `Write Pipeline Data` consist of the barcode number of the module/hybrid, the mode (peak/dec) and the inverter setting (on/off), e.g.

```
Pipe_30216630200048_DecIoff.dat
```

Note: After a pipeline scan, you will probably see a modulation in the height of the pedestals (channel view) with a period of 8 and a magnitude in the order of 1 or 2 ADC counts. This phenomenon is known but unfortunately not yet understood. This can be either problem of the APVs, the MUX or the ARC readout.

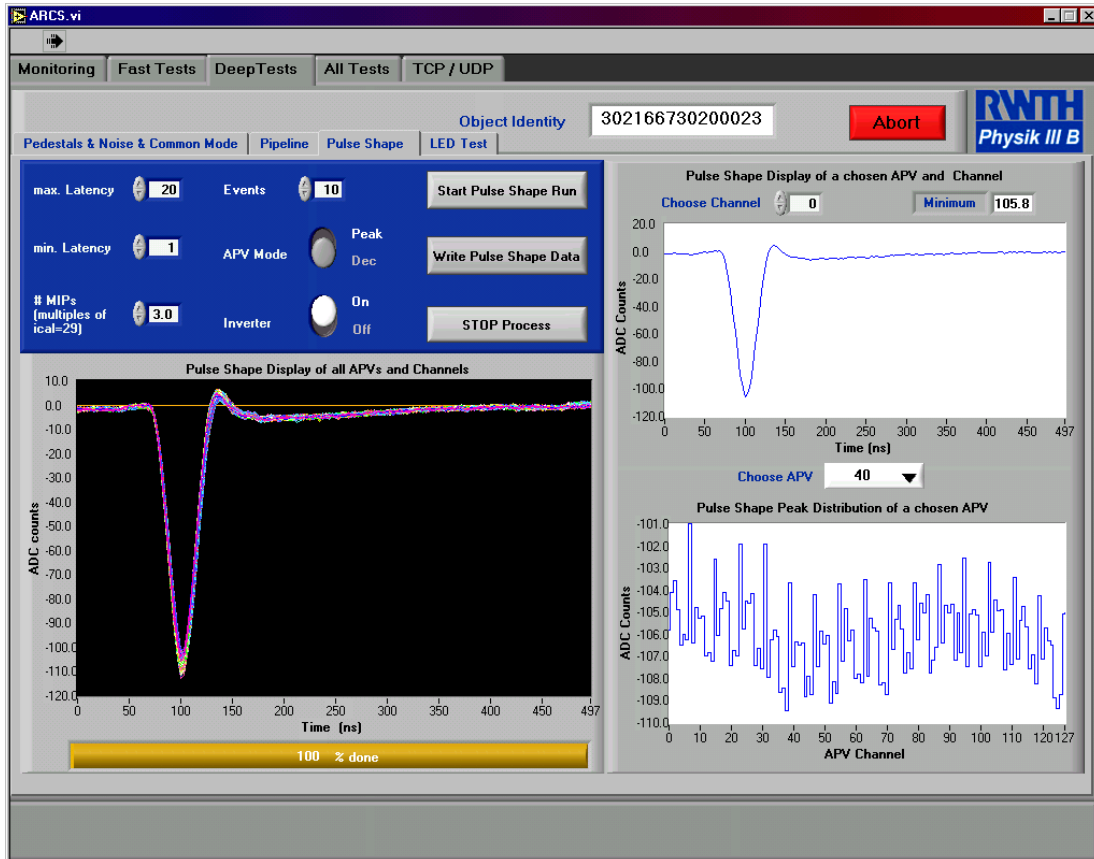


Figure 5.4: The user interface for the Pulse Shape Test.

5.3 Pulse Shape Computation

This feature is foreseen to make pulseshape scans. You can access several registers that influence the shape. These registers are:

- latency you can determine the minimum and maximum latency (use the textfields minlatency and maxlatency). The difference between both can be 19 at most.
- ical the ICAL register is accessed via the textfield mips. “One mip” corresponds to a register value of 29.

mode you can choose between **peak** and **deconvolution** mode and switch the **Inverter on** or **off**. The corresponding values of the mode register are shown below:

peak	inverter ON	mode = 43
peak	inverter OFF	mode = 11
dec	inverter ON	mode = 33
dec	inverter OFF	mode = 1

Note: If the inverter is ON, the VPSP register is set to 25. If the inverter is OFF, the value of VPSP remains as set in the APV_Register.vi (see section 8.1). VPSP = 40 is default.

Note: All other register settings remain the same as they are set using the APV_Register.vi (see section 8.1). That allows to access other registers that influence the shape (e.g. VFS, VFP).

Procedure: During the pulse shape scan, the trigger sequence

$$\underbrace{11000000}_{\text{highbyte}} \quad \underbrace{5}_{\text{spacer}} \quad \underbrace{00010011}_{\text{lowbyte}}$$

is sent to the APVs.

If we call N_{events} the number of events chosen in the control field, then following registers are changed during the scan:

cdrv	every (N_{events})th event
csel	every ($8 \cdot N_{events}$)th event
latency	every ($64 \cdot N_{events}$)th event

The Number of events can be determined using the correlative button. The sequence starts after pressing the button . During the measurement you can see the progress in the Pulse Shape Display of all APVs and Channels, which shows the shape curve for all channels of all APVs. Additionally you can look at the progress bar.

The measurement takes a little longer than in ARCS 4.0, since after every change of the latency a new baseline for that latency is calculated.

After the measurement you can display single channels in the upper right display. The lower right display shows, for each APV, the maxima of the pulseshape curves as a function of the channel. The pulseshape data of all APVs and all channels can be written to an ASCII file using the button. The name of that file consist of the barcode number of the module/hybrid, the mode (peak/dec) and the inverter setting (on/off), e.g.

PSh_30216630200027_DecIon.dat

Note: In spite of some simple corrections on the pulseshape (common mode subtraction, update of baseline) the shape can sometimes show some discontinuities at multiples of 25 ns, especially for modules. This effect is strongly dependent on the grounding you use for the ARC frondend, the HV and the carrier plate. The effect usually decreases a lot when you use thick short cables for the grounding and refer all to one “grounding point”.

5.4 LED Test

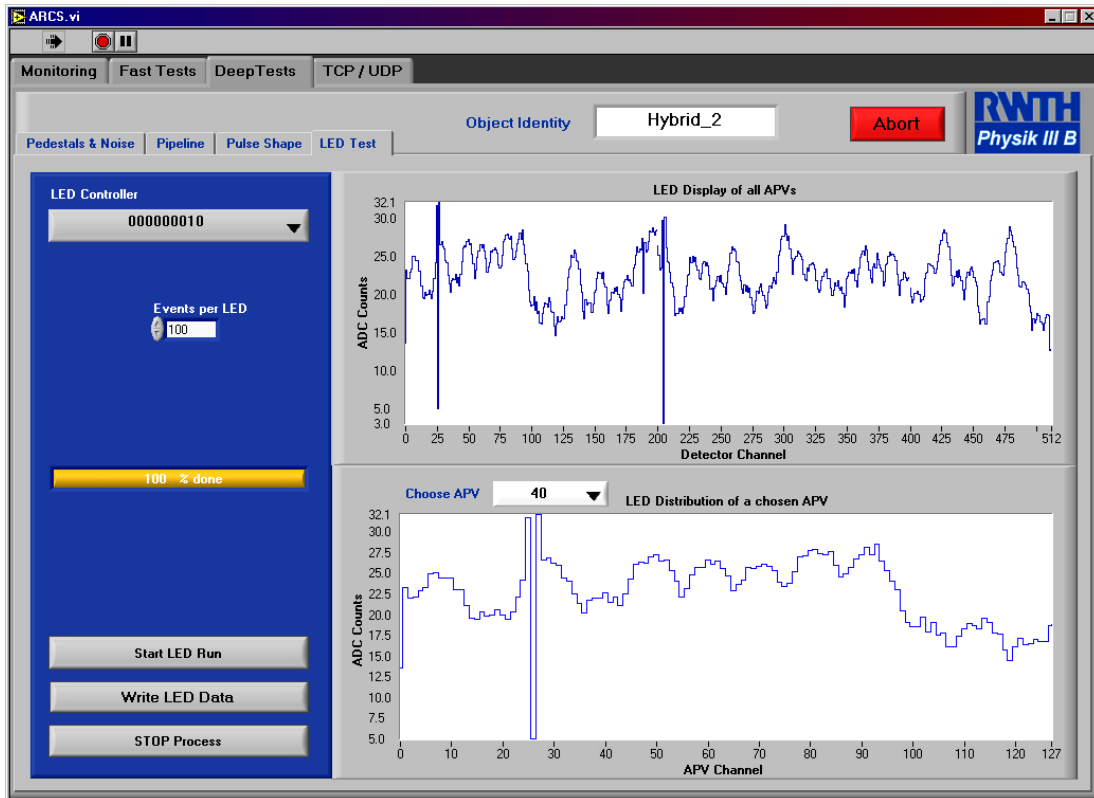


Figure 5.5: The user interface for the LED test.

This test is another new feature of ARCS 5.01. It provides an automated LED test but requires the LED Controller LEP 16 (see section 8.5). Additionally the trigger output of the LED controller must be connected to the trigger input of the ARC board. All other settings (auto repeat mode for the LED controller, external trigger for the ARC board etc.) are automatically done by the test.

One of 16 LEDs is switched on for a certain number of events (which one can set via the textfield `Events per LED`). Then the next LED is switched on for the same number of events, after the previous one has been switched off. When the next LED is switched on, a new calculation of the pedestal (with the LED signal) for each is started. In the upper right display the maximum of these pedestals is shown for all channels and all APVs.

The LED test is started by pushing the button `Start LED Run`. The upper right window will be updated during the test. The test can be stopped via the `Stop Process`. When the test is finished, the lower right display shows the maxima distribution as a function of channel for a single APV.

When you push button `Write LED Data`, you create a file which name consists of the barcode number of the module/hybrid, the mode (peak/dec) and the inverter setting (on/off), e.g.

```
LED_30216630200056_PeakIoff.dat
```



Further automated test procedures are possible. Suggestions are always welcome!

Chapter 6

All Tests

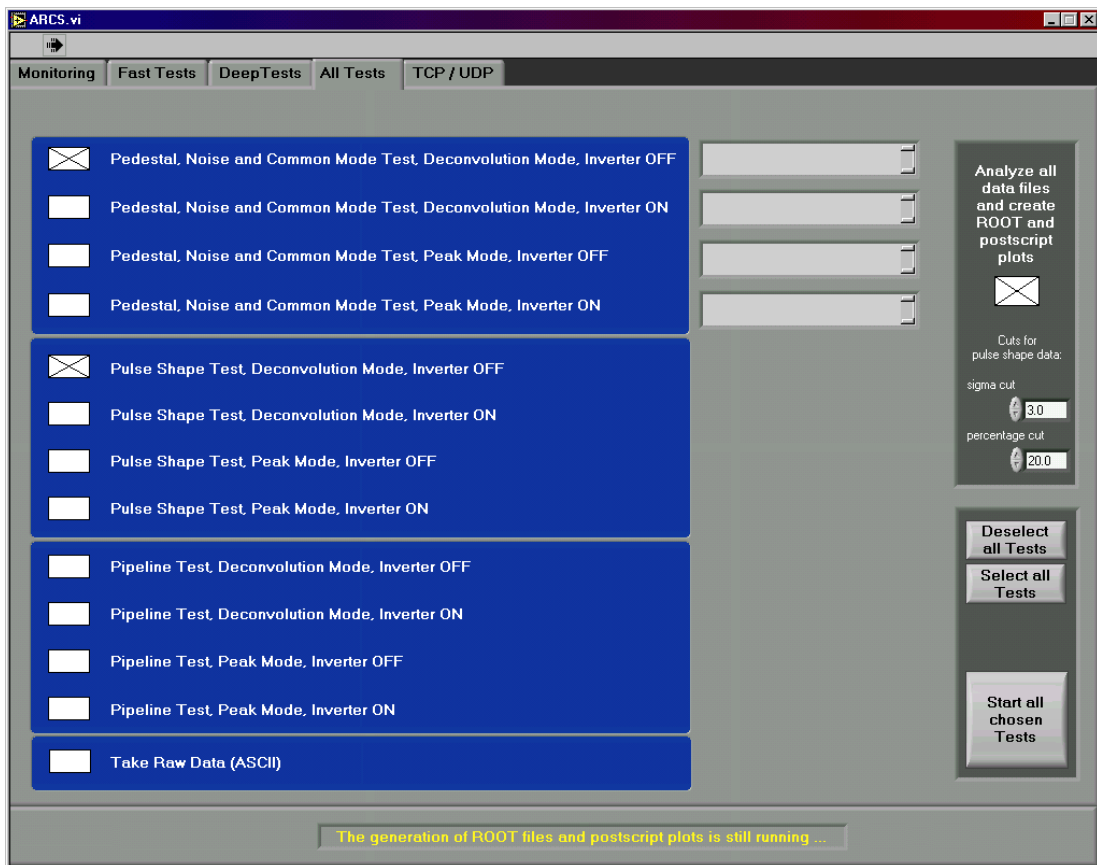


Figure 6.1: A screenshot of the *All Tests* index card.

This index card was added to ARCS 5.01 quite lately. It arose from the request for a tool to perform all the tests successively.

If you want to perform all or none of the tests you can select or deselect all tests by pushing the button or respectively. After a test run ROOT and postscript files will be created if you tick off the corresponding checkbox.

On the right side of the “*Pedestal, Noise and Common Mode Test*” choice field you find four textfields that display the channels flagged as bad during the “Pedestal, Noise and Common Mode Test”.

You can start the sequence of tests with the given test parameters (number of events etc.) by pushing the button .

Note:

Since it uses the tools from other index cards (mainly the Deep Tests) we did not want to cause confusion by adding the same parameter fields again on the All Tests card. So all parameters used for the several tests have to be set on the corresponding panel.¹

The number of events for taking raw data has to be entered on the index card *Monitoring* → *Data* → # of events.

While the test sequence is running no other function of ARCS is accessible.

When the test sequence starts the grey bar below the ARCS screen (the *All Tests* bar) will display a progress bar indicating the advance of the test sequence, the name of the test and the number of tests performed so far. When you push the button the test sequence will be stopped *after* the presently running test. This button is the only accessible button while the test is running.

When you chose to create ROOT and postscript files all other indicators and buttons on the *All Tests* bar will disappear. A note concerning the parallel action of the ROOT and postscript file generation will be displayed instead, as long as this generation lasts.

¹ We know, that this is a little bit circumstantial, but we think it is a useful tool nevertheless. In a later version one can imagine to read the necessary parameters from a settings file to have comparable settings within all tests. All comments and improvement suggestions are welcome.

Chapter 7

TCP/UDP Connection

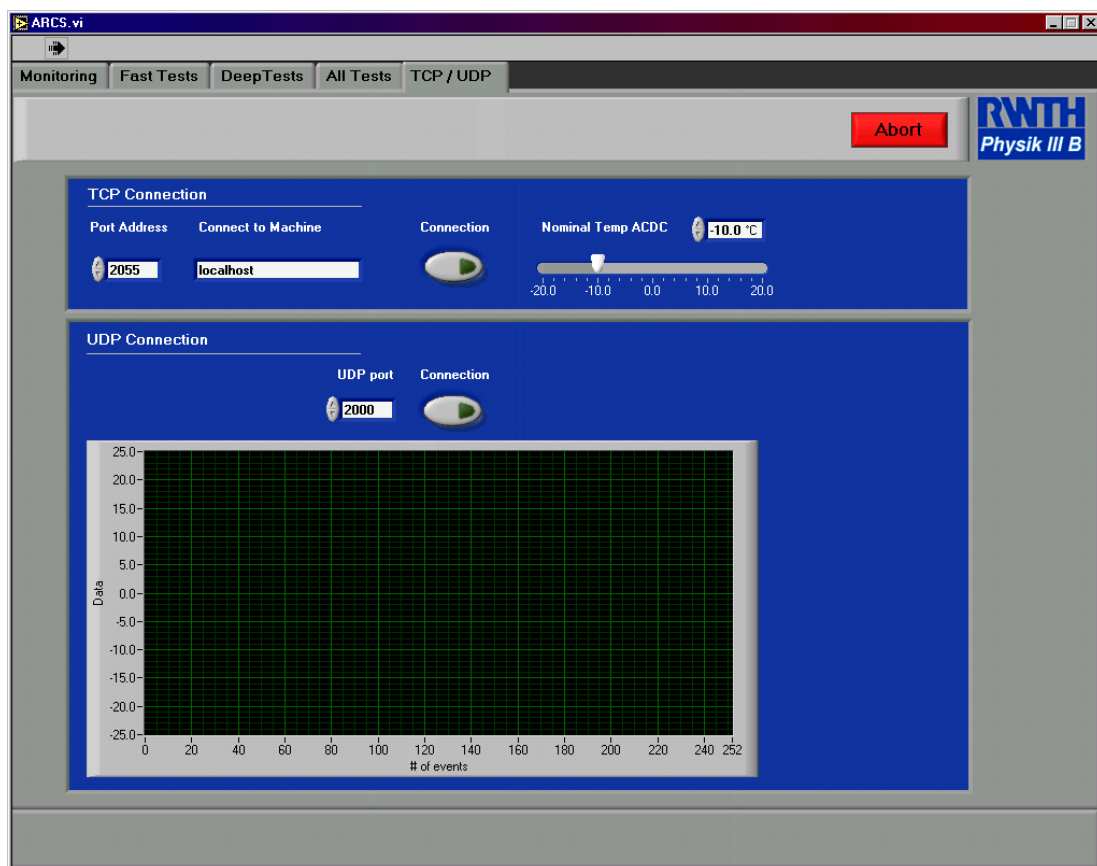


Figure 7.1: A screenshot of the TCP/UDP indexcard.

To have control about temperature and humidity of the cooling box via the ARC software, a TCP/UDP connection to ACDC or any other program, that uses a defined data type scheme

(see section 7.3), can be established. This connection guarantees, that even in case of a unexpected shutdown of the ARC software or the PC, it's running on, the temperature and humidity parameters are under control. Another reason for this feature is the possibility, to integrate other software applications, that are able to send data to the ARC software via TCP. In the current version of ARCS, a TCP connection is used to transmit the nominal temperature to ACDC. Because it's important, that every temperature value is received by ACDC, the TCP protocol is used for this purpose. On the other hand, ACDC sends his monitoring data of all connected sensors via the UDP protocol to ARCS. Therefore it's not possible to receive data, that was measured before a UDP connection was established. Because it's not important, that every single monitoring data is received, a UDP connection is fully sufficient. A UDP connection is comparable to a radio transmitter/receiver unit. Either you have turned on your receiver to get actual data, or you have turned off your receiver. In a TCP connection, the transmitter sends data until the receiver sends an acknowledge. In this case it is guaranteed that the sent data is received.

7.1 TCP Connection

7.1.1 Sending nominal temperature via TCP

To send the nominal temperature via TCP to the ACDC (or another program), you just have to choose a port, fill in the "network name" of the transmitting PC and press the button. If the other program listens on the same port and receives data, the connection button will remain in the pressed state, other way, it will turn off again. In this case check the right port, the file format and the state of the receiving program. If the connection is stable, you can control the nominal temperature via the horizontal pointer slide.

7.2 Receiving Data via UDP

Receiving Data via UDP is quite simple. You just have to adjust a free UDP port and press the button. If everything works, you should see a plot legend with the connected sensors right beneath the waveform graph and, of course, the transmitted data. If you use ACDC, a star beneath one sensor indicates, that this sensor is the regulation sensor. If nothing appears, check the UDP ports and the data format.

7.3 Data Format

As mentioned before, ARCS expects a special data format resp. kind of data for the TCP and UDP connection.

Kind of Data	Format
number of sensors	4 byte int
temperature data of the sensors + peltier voltage	4 byte float per sensor
string with sensornames	tab separated spreadsheet string

Table 7.1: Data String for UDP Communication

7.3.1 UDP Data Format

The expected data for the UDP communication has the format of a string, which consists of three elements: Number of sensors, Data and Data Description. For details see table 7.1.

7.3.2 TCP Data Format

The expected data format for the TCP communication is 4 byte float, namely the nominal temperature value.

Chapter 8

VIs

A VI in the meaning of the LabVIEW programming language is a “Virtual Instrument” containing buttons, digital or analogue displays and different kinds of controllers.

In this chapter those VI’s are explained that you come in contact with while using ARCS.

From ARCS Version 4.0 on, some important VI’s can run in parallel to the main ARCS VI. This remedies the unhandsome fact of previous versions where these VI’s had to be closed before a change in the display could become visible.

8.1 APV_Register

When this VI is called for the first time, the recent values of the APV registers are read from the actual hybrid. “The actual hybrid” means in that context the hybrid you chose to be displayed in the monitoring window. If you run more than one ARC board with ARCS, the actual hybrid also depends on the board you chose to be displayed. You can check and change this using the menu Hybrid ? ▼ in the upper right corner.

On the left side the names of the different registers are displayed. The value of each register is shown in the digital display and the sidebar at the right of each name. If you wish to change some settings you can either click on the slidebars or use the arrow keys beside the digital display. The representation of the numbers in the digital display can be decimal or hexadecimal. Keep in mind that not every setting is senseful.

Register	Range
Mode	0...63
Latency	1...190 (values of 0 or 191 do not result in reasonable data)
Muxgain	0...31 (only one bit should be set)
other registers	0...255

Loading of new settings is done via the “Transfer” buttons. You can either load the same

settings to all APVs at once (use **Transfer to all addresses**) or load settings separately to each APV (use **Transfer to single address**). The address is corresponding to the one chosen as “Device”). Immediately after a transfer all registers of the corresponding APV will be read again and their values will be compared to the values that should be written. If any difference occurs for a certain register, the light at the right side of the register slide bar will turn from green to red. If you transfer new register settings to all APVs at once, the registers of all APVs will be read back and compared. In that case a difference for any register and any APV will result in a red light.

If you once found settings that are reasonable for your test, you can save these settings using the **Write Values to File** button. Later you can load these settings (**Load**). In both cases a browser goes into action.

If you **Set Default Values** the file `APVdefault.set` will be read from the ARCS directory.

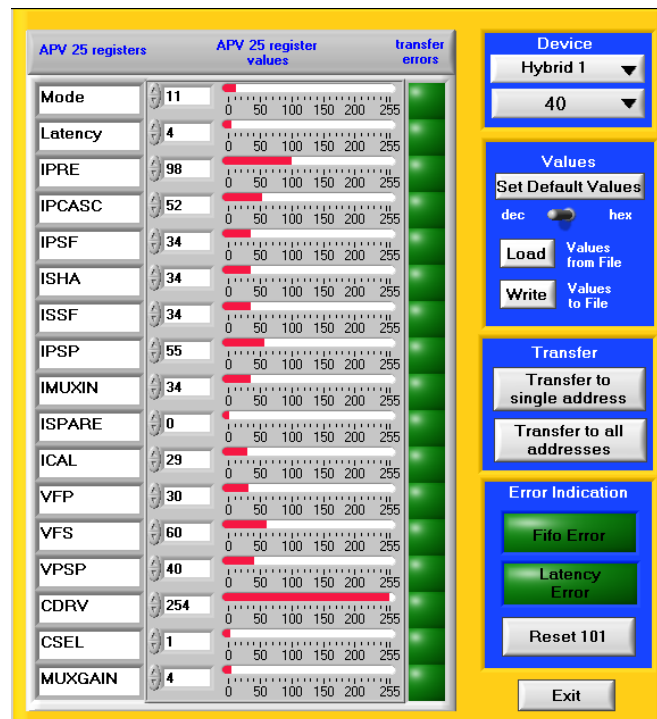


Figure 8.1: A screenshot of the APV_Register.vi used to read recent APV settings and to load new ones in the registers of certain APVs

The Error Indication field serves as a decoding of the Error Register of the APV. Both kinds of errors (Fifo and Latency) normally can be cleared with a **Reset 101**.

8.2 Trigger

This VI is used to load different trigger patterns into the trigger sequencer of the ARC board. The sequence itself consists of a high and a low byte with spacer in between. The number loaded in the spacer gives the number of zeros in the pattern between high and low byte. So if your trigger pattern is composed of (less than) 16 bits you do not need the spacer (i.e. spacer should contain a “0”). If you need a pattern with more than 16 digits you have to split it in two 8 bit parts divided by the spacer.

Example (to access a certain pipeline cell)

$$\underbrace{101000000}_{\text{highbyte}} \underbrace{0\dots\dots 0}_{\text{spacer}} \underbrace{00000001}_{\text{lowbyte}}$$

If the spacer is equal to 5, the full trigger sequence is

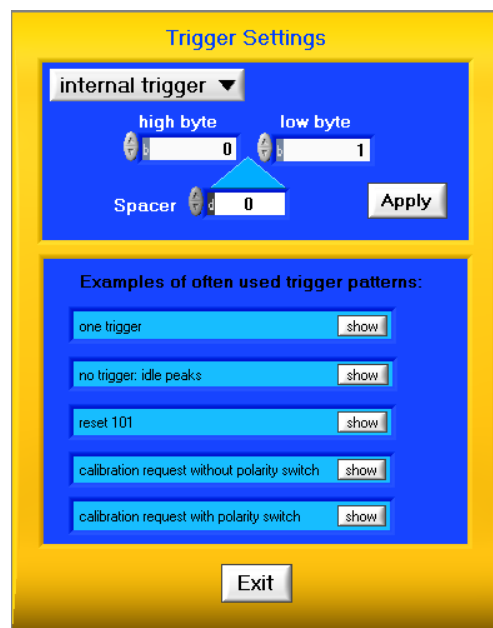
$$10100000 00000 00000001$$


Figure 8.2: A screenshot of the Trigger.vi used to program the Trigger sequencer of the ARC board

One can choose between internal and external trigger.

Note: An internal trigger is generated by the software as soon as one readout cycle is finished. An external trigger can not be accepted before a readout cycle is finished. If a trigger signal is sent to the TRG plug of the ARC during the readout cycle it will be ignored.

Every change of the trigger pattern has to be finished with the **Apply** button.

8.3 DAC Offsets

The data of two neighbouring APVs comes as analogue data from the Hybrid on one output line (multiplexed), is amplified on the Front End Adapter and finally digitized in FADCs on the ARC board. To adapt the analogue signals to the working range of the FADCs an offset can be added to the signals via some DACs. This guarantees the usage of the full FADC range of 8 bit. Of course you have to press **Apply** to update your changes.

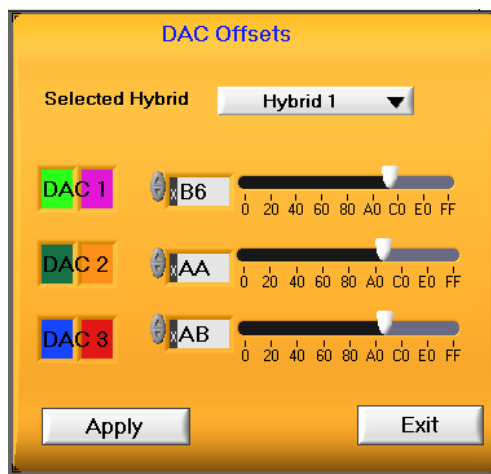


Figure 8.3: A screenshot of the DAC_Offsets.vi used to adapt the analogue signal levels to the working range of the FADCs

8.4 PLL Control

When this VI opens, a register read sequence is done at first to load the current register settings. These values and all values read back later, will be displayed in the right column of the corresponding registers (textfield with white background).

Switching the *mode* register of *Control and Status Register 1* to **auto calibration ▼** (default setting) disables all choice buttons which allow to set further chip registers. The set of values while working in auto calibration mode can be read back using the **read all registers** button. Switching the *mode* register to **test mode ▼** enables all choice buttons for further access to the PLL chip. Choosing several values is done via the choice buttons. To write these values to the PLL chip, push the **write all registers** button. A detailed description of the purposes of the several registers can be found in the CMS Tracker PLL Reference Manual.¹

¹<http://cmstrackercontrol.web.cern.ch/CMSTrackerControl/documents/PauloMoreira/TrackerPLLManual.pdf>

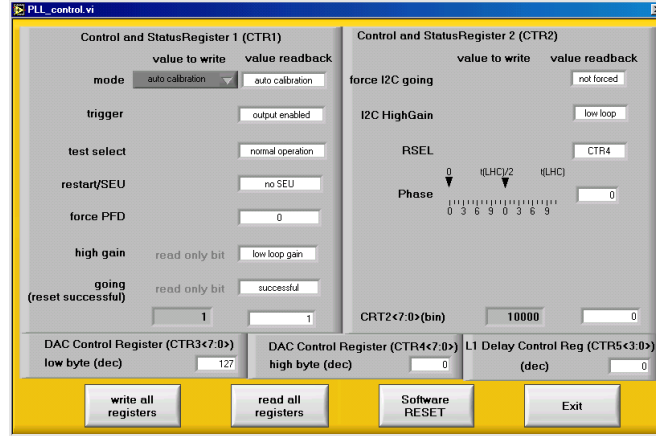


Figure 8.4: A screenshot of the PLL_Control.vi. This VI allows read/write access to all PLL registers

For a basic control of write and read sequences, the belonging bit patterns are additionally displayed underneath the Control and Status Registers CTR1 and CTR2.

Note: The registers CTR1<5:4> are summarized in one choice button - *test select*. The registers CTR2<4> and CTR<3:0> are summarized in one slide controller - *Phase*. The *RSEL* choice in the present release has no consequences on the registers to be written to.

write all registers	writes all the registers as displayed (i.e. as the user entered it at last) in the textfield.
read all registers	reads all PLL registers and displays the results in the textfields belonging to the registers CTR1 to CTR5. This allows to control whether the write procedure was successful.
Software RESET	a software reset is done by writing a 0-1 sequence in CTR1<3>; all of the registers are turned to their default values; these values may be read back via the button read all registers .
Exit	closes the PLL dialog window.

8.5 LED Controller

The LED_Controller.vi that opens by pressing the LED button on the *Additional Controller* index card allows to control all functions of the LED controller LEP16.²

²<http://www.physik.rwth-aachen.de/group/IIIphys/CMS/tracker/en/silicon/ledpulser.html>

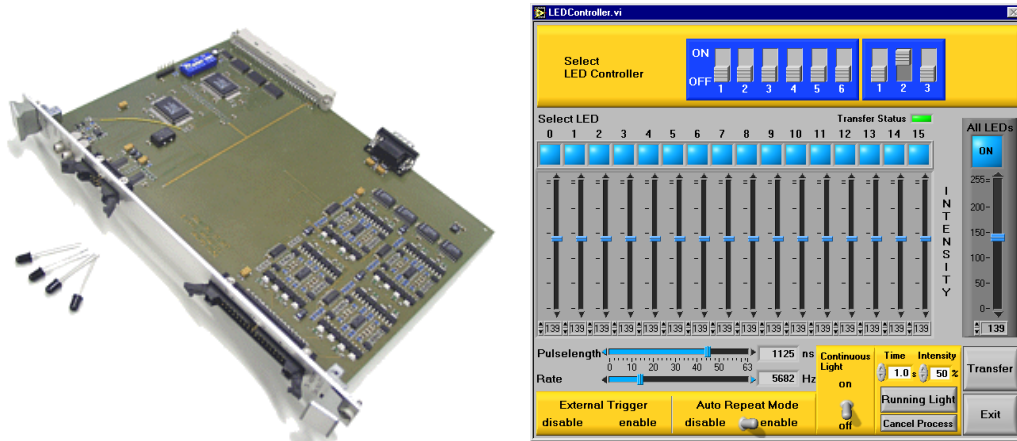


Figure 8.5: A picture of the LED Pulser LEP16 and a screenshot of the LED_Controller.vi. This VI allows to control all functions of the LED pulser

- Select LED Controller* The meaning of the DIP switches is the same as for the ARC boards. You can choose any address as long as this address is not the same as on another board and as long as you do not get in conflict with hardware addresses on your PC.
- SelectLED*
0 1 2 3 ... The lights below the numbers are indicators and switches at once. They indicate which LEDs you select to be switched on. Their state can be changed by a simple mouse click. The intensity can be set in 256 steps with the slidebar below the indicator light.
- All LEDs* If you want to switch on/off all LEDs at once or change the intensity of all LEDs in one step you should use the light switch and intensity regulator on the right margin below the text “All LEDs”.
- Pulselength* This slidebar is used to determine the length of the *electrical* pulse the LEDs are supplied with in multiples of $25ns$ (the light pulse is slower in rise- and falltime).
- Rate* determines the rate (in Hz) in the auto repeat mode. 512 steps are possible.
- External Trigger* With this switch you can enable/disable the effect of a trigger signal coming in through the TRG IN of the LEP16 controller. This switch vanishes if the “*Auto Repeat Mode*” is enabled.
- Auto Repeat Mode* The purpose of that switch is to enable/disable the auto repeat mode logic. If it is enabled, trigger signals with the chosen auto repeat rate (see above) are sent to the TRG OUT plug of the controller. This switch vanishes if the *External Trigger* is enabled.

Running Light

Pushing this button starts a sequence in which each LED is switched on for a time (that can be determined in the textfield above the button), then the next one and so on.

Transfer

Pushing this button writes the values you set for the board you choose into its registers.

Exit

Closes the LED_Controller.vi window.

8.6 DCU Test

This VI is an approach of a LV and DCU test. The idea is, that changes of the voltage the hybrid is supplied with, should be detected by the DCU voltage monitoring channels. So you

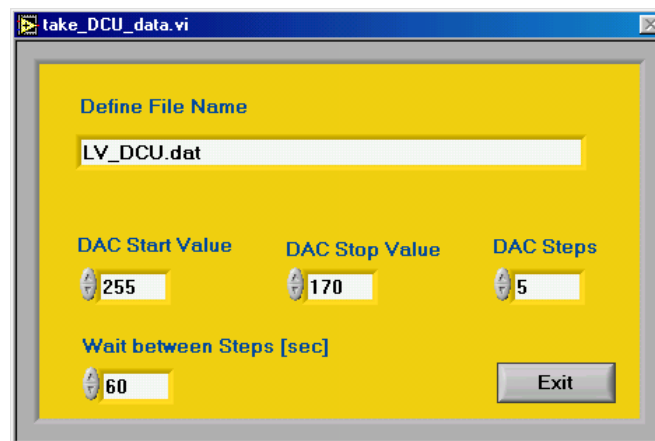


Figure 8.6: A screenshot of the take_DCU_data.vi used to drive the voltage via the ARC Frontend Adapter and compare changes with the DCU voltage measurements

can determine the voltage, the hybrid is supplied with by the frontend adapter. A DAC value of 255 corresponds to 2.63 V/1.3V on the V250/V125 line and a DAC value of 0 corresponds to a 2.12V/1.06V. That means $\approx 2\text{mV}/1\text{mV}$ per DAC count. Since voltages below 2.45 causes problems in the I²C access to the chips, lower limits of less than 170 are not recommended.

The test sequence starts at the *DAC Start Value* and changes in steps which size is set via the *DAC Steps* textfield. The sequence stops, when a DAC value is equal or less than set as *DAC Stop Value* (which should be lower than the start value). Since the electronics needs some time to set new values and read back stable data, one should wait some seconds between setting a new value and reading the measurement of the DCU. This delay can be set using the textfield *Wait between Steps [sec]*.

The sequence starts when you push the **Exit** button. It can be stopped at any point of the

sequence via the **Write LV, DCU Data** button (which you pressed to start the DCU test VI). The status of this test can be seen on the LV and DCU supervisions (see section 3.2). During the data taking phase, you should observe the automated change of the LV switch for changing the hybrid voltage supply. It's worth to compare the DCU output with the LV displays (see figure 8)!

8.7 Analysing ARCS data with LabVIEW2Root

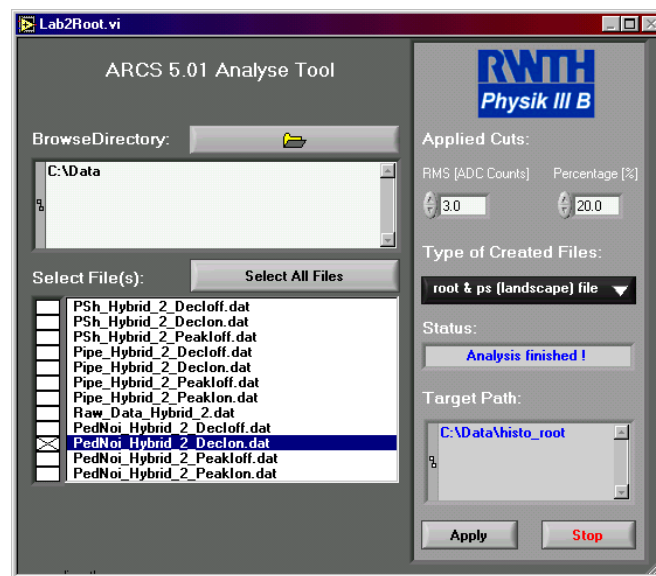


Figure 8.7: A screenshot of the LabVIEW2Root window.

ARCS 5.01 offers the possibility to create postscript plots from all ASCII files you originated previously with ARCS. Since we use ROOT (Version 3.01/06 for Windows) tools to achieve this, ROOT files are originated as well. These Root files include sometimes more information than the postscript files which are intended to give a comfortable overview of the test results.

To choose the directory push the corresponding button *BrowseDirectory* and browse to the directory containing your data files (press button **Select Cur Dir** to approve your decision). Once a directory path is chosen the files which can be analysed are displayed. Marking the files (per default all files are marked) and pushing the **Apply** button starts the creation of root and postscript files using the *lr.exe* application. Afterwards you can find these files in a subdirectory of the chosen directory path. The subdirectory is called *hist_root*. In the window *Target Path* the path of the directory containing the created files is given.

Note: Since ARCS 5.01 a result file is created to summarize the most important results of the

different data files. This file is called `result.log` and includes mean values, rms values and bad channel lists. The meaning of these values is self-explanatory.

The format of the postscript files can be determined by using the `root ... file ▼` button. In addition you can apply two different cut strategies to the data: a cut on the rms and a percentage cut with regard to the mean value. The cut algorithms are implemented as follows: First of all the mean value of the complete data sample is calculated and the most conspicuous channels are marked via a $\pm 3 * rms + mean$ cut (the values are called *mean* and *rms* in the `result.log` file). Neglecting these marked channels new mean and rms values (called *tmean* and *trms* in the `result.log` file) are computed. Afterwards your chosen cuts are applied to the data (e.g. 2.0 in the RMS box means a cut of $\pm 2.0 * trms + tmean$ and a 20.0 in the percentage box means a cut of $(\pm 0.2 + 1) * tmean$).³

It is not necessary to halt the DAQ during the analysis is ongoing.

³The only data files not being analysed this way are called *PedNoi_?.dat* and *Pipe_?.dat*. Here Valery Zukhov's calculation methods are used for the cuts.

Index

- All Tests, 28
- APV & Boards, 10
- APV Properties, 11
- APV-Display, 9
- APV_Register, 33
- Automated Fast Test, 15
- Board Properties, 10
- Common Mode Distribution, 20
- Configuration File, 4
- DAC_Offsets, 36
- Data, 12
- Data Format, 31
- Data Panel, 13
- DCU Supervision, 13
- DCU Test, 39
- Deep Tests, 18
- Errors, 11
- Hybrid Supervision, 12
- Index Cards of the Monitoring Supervision,
9
- initialization, 5
- Initialization Phase, 4
- Invocation, 16
- LabVIEW2Root, 40
- LED Controller, 37
- LED Test, 25
- LEP 16, 37
- Monitoring-Phase, 8
- MUX & PLL, 11
- MUX Control, 11
- Noise Calculation, 19
- Pedestal Calculation, 19
- Pipeline Scan, 20
- PLL Control, 11, 36
- Pulseshape, 23
- Receiving Data via UDP, 31
- Running Automated Tests, 16
- Sending nominal temperature via TCP, 31
- TCP Connection, 31
- TCP/UDP Connection, 30
- Trigger, 35
- VI explanation, 33