The ATLAS Barrel Alignment Readout System

Reference Guide

Robert Hart
NIKHEF, Amsterdam, Netherlands

Abstract

The barrel alignment system of the ATLAS muon spectrometer consists of 5812 optical lines, each one built up of a camera, a light source, a coded mask and a lens. Three layers of multiplexing are applied, controlled by eight PCs, each one equipped with a frame-grabber, which acquires the images to be analyzed. The analyzed results are stored into a database for off-line track reconstruction. The multiplexers and frame-grabber are controlled by the Rasdim server, which itself is used by the PVSS SCADA system. This document describes the setup of the system in USA15 and how it is used by PVSS. It also contains instructions how to setup and debug the system.

Contents

1 General Description

The optical lines (also known as *channels*) of the barrel alignment system are mounted in the ATLAS cavern, mainly attached to the MDT chambers. A three layer multiplex-scheme (TopMux → MasterMux → RasMux) connects the channels to the eight TopMuxes located in USA15. Eight PCs, equipped with a frame-grabber, are utilized to control the readout, which in general means:

1. Select channel
2. Grab image
3. Analyze image
4. Store results into database

For each channel this sequence is one after the other repeated in an infinite loop.
The channels are split into eight groups, each one controlled by its own TopMux/PC combination as shown in the logical diagram of Figure 1.1.

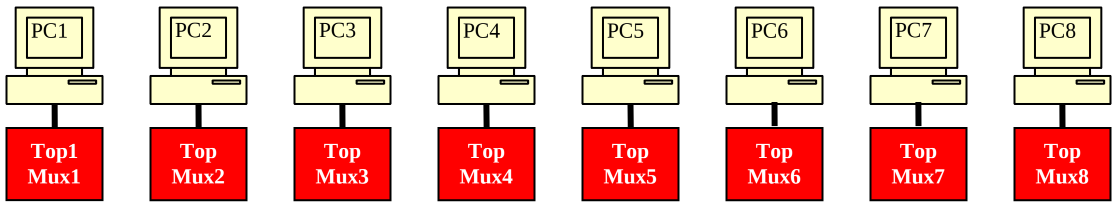
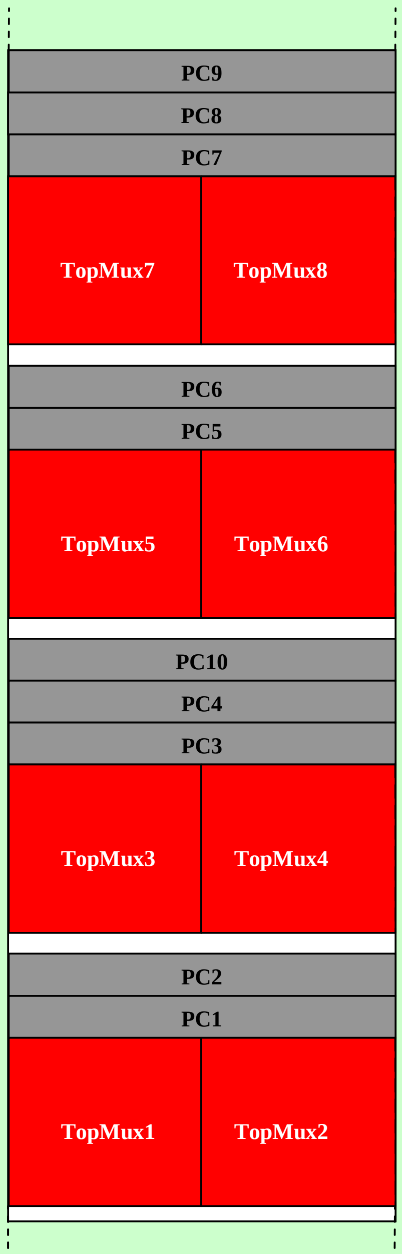
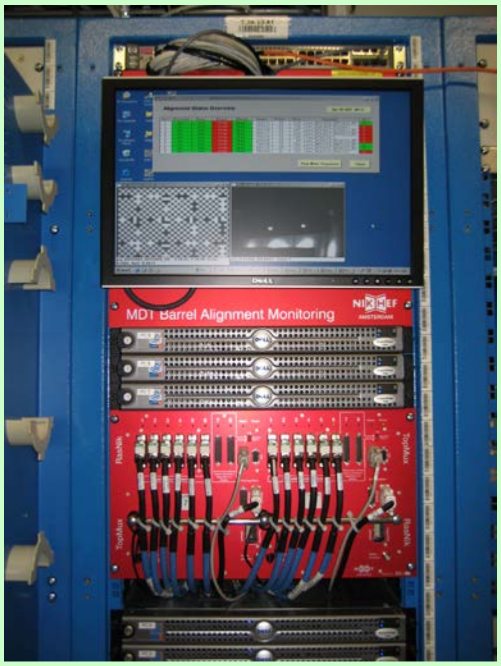


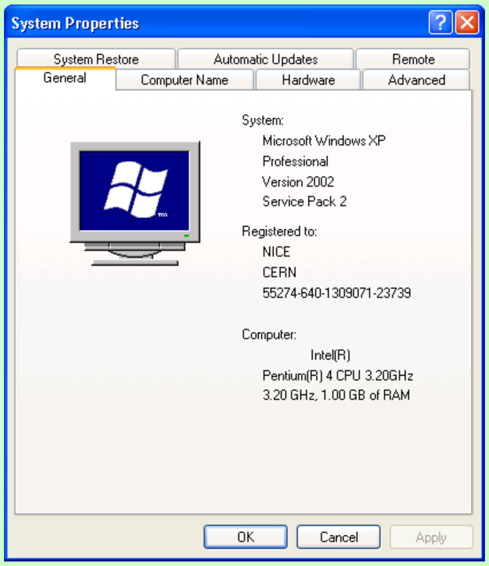
Figure 1.1: Eight PC/TopMux combinations

1.1 USA15

The hardware belonging to Figure 1.1, PCs and TopMuxes, is mounted in a single 19’’ rack **Y.28-19A1**, located in USA15. A physical layout of the applied hardware is shown in Figure 1.2. The drawing left is incomplete and not to scale. It only shows the PCs and TopMuxes in their relative positions. The photo in the upper right corner shows a part of the rack. There are ten PCs. Eight of them are, as mentioned before, used to control the TopMuxes and are known as production PCs. PC9 is used as super-visor, monitoring the other PCs and holding the FSM part. PC10 is used as spare, to be used whenever one of PC[1-8] fails. For that purpose it is located in the middle of the rack, so each cable is able to reach PC10. The PCs are rack-mountable and 1U high. The TopMuxes are mounted by pairs of two in a 6U high panel. Beneath each TopMux a gap of 1U high is maintained in order to connect the cables to the MasterMuxes in the cavern.
Other hardware mounted in the rack: in top an Ethernet switch, in the middle a 1U high retractable LCD-screen with keyboard and trackball, a drawer and in top at the back a KVM-switch to which all PCs are connected.
Rack **Y.28-19A1** is powered by **UPS** and is horizontally cooled by means of big fans mounted in the rear door.







System:
Microsoft Windows XP
Professional
Version 2002
Service Pack 2

Registered to:
NICE
CERN
55274-640-1309071-23739

Computer:
Intel(R)
Pentium(R) 4 CPU 3.20GHz
3.20 GHz, 1.00 GB of RAM

Figure 1.2: Front-view rack and System Properties

The manufacturer of the alignment PCs is *DELL*, type PE750, running Windows-XP, 3.2 GHz, 1 GB of RAM, 80 GB hard-disk, rack mountable, 1U high. It does not support IPMI. The system properties are shown in the lower right corner of Figure 1.2. It has one PCI-slot (5 Volt), which is used for the frame-grabber and it has an Ethernet connection. The keyboard, mouse and VGA-output of the ten PCs are connected to the KVM-switch, which itself is connected to the retractable video-screen with keyboard and trackball.
As frame-grabber the DT3162 of *Data-Translation* is used. It is a monochrome frame-grabber for the PCI bus. PC[1-8] and the spare PC10 are equipped with one, occupying the single PCI slot. PC9 is **not** equipped with a frame-grabber. Version 1.1.2.3 of the frame-grabber's driver is installed.

The TopMuxes are mounted in pairs on a single 6U high frame. All connectors are at the front side. Eight connectors are used to be connected to the MasterMuxes. Only outlets 1 to 6 are used. Outlets 7 and 8 are not used (yet) and may also be used to be connected to a RasMux. Normally the outlets can be connected to MasterMuxes only. However if the jumper at the front is set to ‘RasMux’, then outlet 7 and 8 may be connected to a RasMux as well. Look at Figure 1.3 for details. Other important connectors are the power, COM-port and frame-grabber. The power of a TopMux (24 Volt dc) is delivered by a ‘simple’ power-converter, which itself is powered by UPS too. The COM-port of the alignment PCs is used to control the TopMux. Figure 1.4 shows the connection diagram of a typical alignment PC.

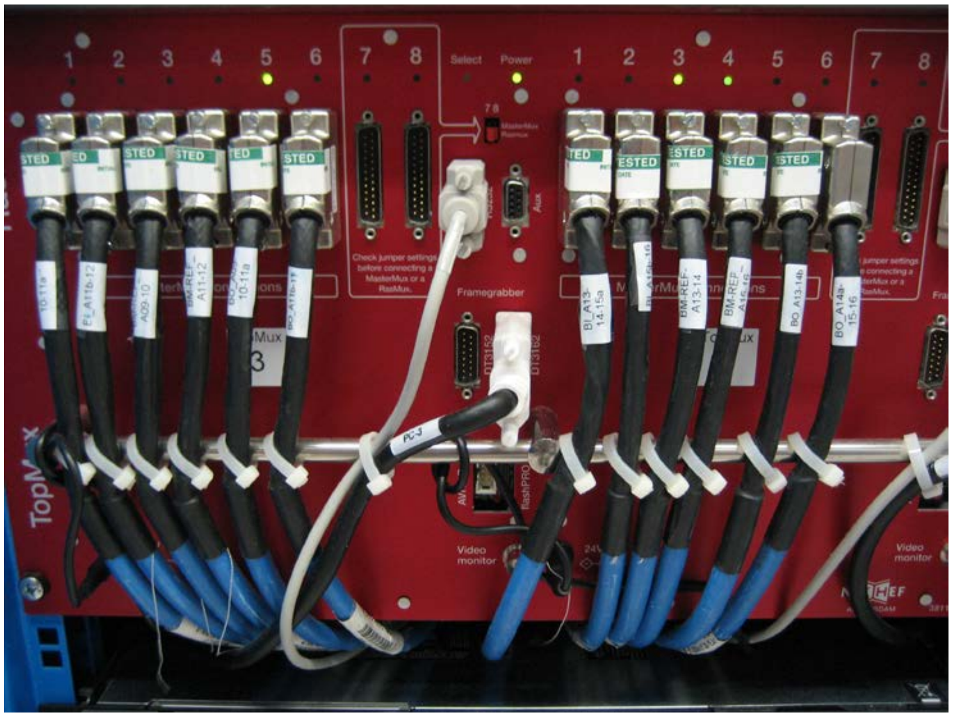


Figure 1.3: TopMux front-view

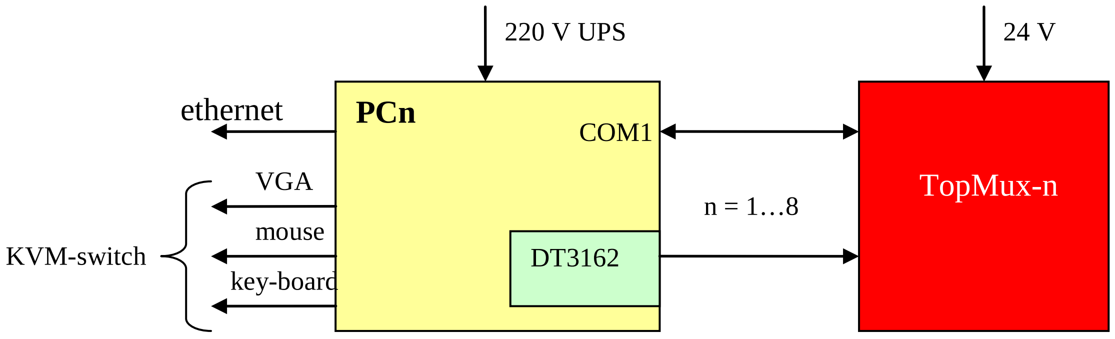


Figure 1.4: PC-TopMux connection diagram

All cables are labeled at both ends, indicating to which alignment PC they belong: i.e. PC1-10. The labels of the cables connecting the MasterMuxes are listed in Table 1.1 for barrel side-A and Table 1.2 for side-C. The names designate the region/section of the connected MasterMux.

Nr.	**TopMux1**	**TopMux2**	**TopMux3**	**TopMux4**
1	BI_A01-02	BI_A05-06	BI_A09-10-11a	BI_A13-14-15a
2	BI_A03-04	BI_A07-08	BI_A11b-12	BI_A15b-16
3	BM-REF_A01-02	BM-REF_A05-06	BM-REF_A09-10	BM-REF_A13-14
4	BM-REF_A03-04	BM-REF_A07-08	BM-REF_A11-12	BM-REF_A15-16
5	BO_A01-02	BO_A05-06	BO_A09-10-11a	BO_A13-14b
6	BO_A03-04	BO_A07-08	BO_A11b-12	BO_A14a_15_16
7	*nc*	*nc*	*nc*	*nc*
8	*nc*	*nc*	*nc*	*nc*

Table 1.1: TopMux labels A-side

Nr.	**TopMux5**	**TopMux6**	**TopMux7**	**TopMux8**
1	BI_C13-14-15a	BI_C09-10-11a	BI_C05-06	BI_C01-02
2	BI_C15b-16	BI_C11b-12	BI_C07-08	BI_C03-04
3	BM-REF_C13-14	BM-REF_C09-10	BM-REF_C05-06	BM-REF_C01-02
4	BM-REF_C15-16	BM-REF_C11-12	BM-REF_C07-08	BM-REF_C03-04
5	BO_C13-14b	BO_C09-10-11a	BO_C05-06	BO_C01-02
6	BO_C14a_15_16	BO_C11b-12	BO_C07-08	BO_C03-04
7	*nc*	*nc*	*nc*	*nc*
8	*nc*	*nc*	*nc*	*nc*

Table 1.2: TopMux labels C-side

The Full Qualified Names (FQN) of the alignment PCs are listed in Table 1.3.

PC	**FQN**
1	`pcatlmdtbal1.cern.ch`
2	`pcatlmdtbal2.cern.ch`
3	`pcatlmdtbal3.cern.ch`
4	`pcatlmdtbal4.cern.ch`
5	`pcatlmdtbal5.cern.ch`
6	`pcatlmdtbal6.cern.ch`
7	`pcatlmdtbal7.cern.ch`
8	`pcatlmdtbal8.cern.ch`
9	`pcatlmdtbal9.cern.ch`
10	`pcatlmdtbal10.cern.ch`

Table 1.3: PC vs. FQN

1.2 Controls

Two types of coded masks are applied: the *Rasnik* type and the *Spot* type, hence for each one a distinct analysis module is available (as dynamic link library). The Rasnik type has a chessboard pattern with encoded edges, the Spot type has a mask with four holes. Figure 1.5 shows an example of the two types. Around 90% of the ATLAS channels are of type Rasnik, the remaining 10% of type Spot.

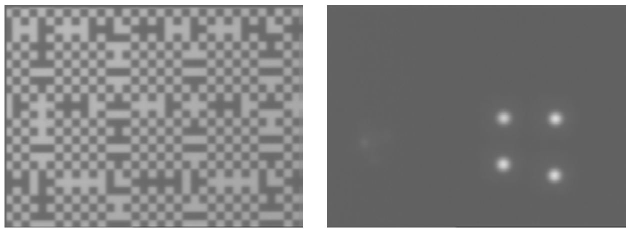


Figure 1.5: Rasnik and Spot mask

Before an image can be taken, the chosen optical-line (or channel) has to be activated, by selecting its light source, its optical sensor (a camera without lens) and optionally (by means of the I^2C bus) some settings like gain and exposure time. When done, the frame-grabber is connected to the channel. To control and use the TopMux and frame-grabber, three components are applied:

1. **Rasdim.** This application (or server) controls the TopMux and frame-grabber. It is developed using Visual C++. At startup it initializes the frame-grabber and resets the TopMux. After a successful initialization, it waits for commands from any client. It should be clear that the Rasdim server controls only the TopMux and frame-grabber of the PC it is running on!
2. **PVVS.** A commercial SCADA product holding the channel parameters. The basic data-container of a variable is called a *datapoint*, which could be everything from being a simple type (integer, float, etc) or a complex type like an array, structure or reference to another datapoint. Its main task is to issue the select/analyze commands to the Rasdim server the channel belongs to. Eight PVSS projects are defined, for each group of channels one. The project runs on the PC/TopMux combination it belongs to.
3. **DIM.** A lightweight client/server communication protocol, providing easy communication between PVSS (or any other client) and Rasdim. It requires that somewhere the dynamic name server (*dns*) is running. For that purpose PC9 (super-visor) is used.

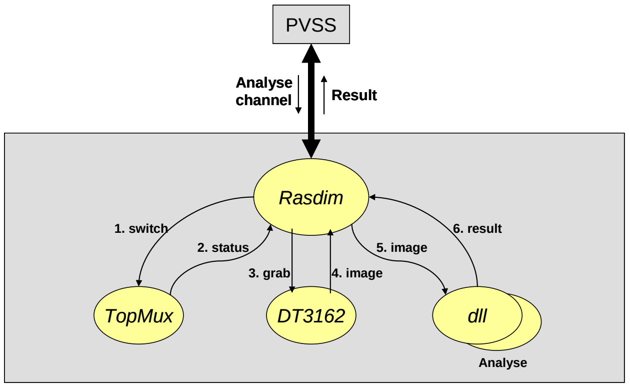


Figure 1.6: Rasdim control flow

Figure 1.6 shows the control flow of the three components. A typical select/analyze channel command triggers the Rasdim application to perform the following steps:

1. Switch the desired led and camera on. Optionally any I^2C setting is set as well.
2. If the previous step was unsuccessful, return the status to the client, otherwise continue.
3. Issue the grab image command.
4. Grab the image.
5. Submit the image to the appropriate analysis module.
6. Fetch the result and send it back to the client.

The result sent back contains the four major identifiers (including their error margins):

1. Translation *x* with respect to the optical axis *z*
2. Translation *y* with respect to the optical axis *z*
3. Magnification of the optical system (or *z*)
4. Rotation angle θ_z between mask and optical sensor

There are eight groups of channels; eight PCs, eight Rasdim servers, eight PVSS projects and all of them have a one to one correspondence. Table 1.4 shows these interrelations, including the number of channels per group, the name of the PVSS project and its PVSS distribution number.

PC	Server	Channels	PVSS-SystemName	PVSS Distr. Number
1	Rasdim1	707	ATLMDTBAL1	91
2	Rasdim2	702	ATLMDTBAL2	92
3	Rasdim3	753	ATLMDTBAL3	93
4	Rasdim4	759	ATLMDTBAL4	94
5	Rasdim5	759	ATLMDTBAL5	95
6	Rasdim6	751	ATLMDTBAL6	96
7	Rasdim7	698	ATLMDTBAL7	97
8	Rasdim8	704	ATLMDTBAL8	98
9	*na*	-	ATLMDTBAL9	99
10	*na*	-	*na*	-

Table 1.4: Server and PVSS attributes

The PVSS distribution numbers are unique within the ATLAS DCS environment. They are allocated by the central DCS-group. The system-name is also unique and by its name it is possible to look at the datapoints of other PVSS projects. Whenever PC10 has to take over PC[1-8], it should have a PVSS project with the distribution number of the one that failed. It should also take over the corresponding Rasdim server.
The addresses of the light-source and camera have the same layout and consist of four items:

1. Server	`[1- 8]`
2. TopMux	`[1- 8]`
3. MasterMux	`[1-16]`
4. RasMux	`[1- 8]`

1.2.1 Mixed-Channels

The addresses of the light-source and camera are completely independent. Only one aspect is mandatory: they should have the same server number, hence they are selected/controlled by the same PC/TopMux combination. In the original setup it was possible to have *mixed-channels*, i.e. the light-source and camera were behind a different PC/TopMux combination. It made the system complex and slow. The only benefit was a simpler cabling scheme. The concept of mixed-channels is abandoned

1.2.2 Analysis

The Rasdim server supports three analysis modules:

1. **`RASNIK`** `(#0):` the original analysis module for Rasnik masks. Written in *Visual-Fortran*, slow, not free of bugs, but still available because sometimes it analyses pictures where others fail. Currently not used anymore.
2. **`SPOT`** `(#1):` the analysis for the Spot masks. Written in *C++*.
3. **`FOAM`** `(#2):` at the moment the de-facto analysis module for Rasnik masks. It consists of a *C++* and *Pascal* part and is compared to Rasnik very fast. Its name is an acronym of FOurier Analysis Method.

1.3 Databases

Two databases, using Oracle, are used for the alignment system. One is the so-called Conditions Database (*CondDb*), which is used to store the results. The other is the Configuration Database (*ConfDb*) which is used to keep the setup of the channels; i.e. the addresses and parameters. The latter database is used to setup an entire PVSS project from scratch. Modifications of the parameters are collected once a day and stored into the ConfDb, in order to keep it as consistent as possible. Accessing the databases is performed by the **CtrlRDBAccess** package; a PVSS CTRL extension for relational database access.

1.3.1 CondDb

The CondDb has the following attributes:

- Database: **atonr_mdt_dcs**
- User: **ATLAS_MDT_DCS_W**
- Password: ****** (classified)

These three attributes provide entrance to the result table with write permission. The password is for obvious reasons encrypted. The result table is called **ICARASDATA** and its layout is described in Table 1.5. The eight PVSS projects store their data in parallel to this project. For efficiency reasons it uses *bound* variables and *bulk* storage as supported by the CtrlRDBAccess package.

Column	Type	Description
MID	int	Primary key: updated by a SQL-sequence (+1)
CHANNEL_NAME	string	Name of channel
SEQ_NR	int	Sequence number
STIME	time	Date/Time of measurement
GLOB_ERR	int	Global error: if $\neq 0$ see Table 1.6
ANAL_ERR	int	Analysis error: only if GLOB_ERR = 4
XVALUE	float	Translation x [mm]
YVALUE	float	Translation y [mm]
SCALEVALUE	float	Magnification
ROTZVALUE	float	Rotation angle θ_z [mrad]
ERR_X	float	Error variance of XVALUE
ERR_Y	float	Error variance of YVALUE
ERR_SCALE	float	Error variance of SCALEVALUE
ERR_ROTZ	float	Error variance of ROTZVALUE
SERVER	int	Server number belonging to channel [1-8]

Table 1.5: ICARASDATA table

Whenever the analysis module fails to analyze the image, the GLOB_ERROR is set to ANALYSIS (#4) and the ANAL_ERROR to a module specific error value. Each loop, handling the channels once, is called a sequence; hence the SEQ_NR variable. It denotes the set of channels belonging to the sequence. After each loop it is incremented by one.
The GLOB_ERR variable is the result of the measurement/analysis. Several errors are possible and could originate from different sources. The Rasdim server is one of them and the most important one. But also the client, in our case PVSS, or the analysis module may set it. Table 1.6 shows the list of possible errors and their meaning.
Common runtime errors are MUX and ANALYSIS errors. Errors of type GRABBER, NOSERVER and TIMEOUT are more serious, because they indicate either a communication problem or crash of the Rasdim server. In that case it will dominate the entire sequence with this error value. The other error types are very rare or never occurred at all.

Synopsis	Value	Source	Description
OKAY	0	-	Okay!
ILLCMD	1	Rasdim	Unknown command
MUX	2	Rasdim	Multiplexor: camera or light source not connected
GRABBER	3	Rasdim	Frame grabber: no grab within timeout
ANALYSIS	4	Analysis	Analysis failed
OUTPUT	5	Rasdim	Problem with output files
WRONGSERVER	6	Rasdim	Server address of camera ≠ this server
ILLBKGRND	7	Rasdim	Background subtracting not allowed with mixed channels
NOSERVER	8	PVSS	Rasdim not running
TIMEOUT	9	PVSS	Na answer from Rasdim within timeout
INTERNAL	10	PVSS	Internal PVSS problem
UNKNOWN	11	*	Unknown (none above)

Table 1.6: Global errors

For statistical purposes another table is maintained. It is called **ALIGN_STAT** and is updated by after each sequence. The layout is shown in Table 1.7. The meaning of the columns is trivial. It contains a summary of the last sequence like start of sequence, end of sequence, number of handled channels, number of errors and specified for each type of error (Table 1.6).

Column	Type	Description
MID	int	Primary key: updated by a SQL-sequence (+1)
SERVER	int	Server number [1-8]
SEQ_NR	int	Sequence number
STARTTIME	time	Start of sequence
STOPTIME	time	End of sequence
SEQ_LENGTH	int	Number of channels per server
HANDLED	int	Number of handled channels
REJECTED	int	Number of rejected channels
FAULTS	int	Number of errors
E_ILLCMD	int	*Refer to Table 1.6 for the description of these variables.*
E_MUX	int	
E_GRABBER	int	
E_ANALYSIS	int	
E_OUTPUT	int	
E_WRONGSERVER	int	
E_ILLBKGRND	int	
E_NOSERVER	int	
E_TIMEOUT	int	
E_INTERNAL	int	
E_UNKNOWN	int	

Table 1.7: ALIGN_STAT table

Each channel has a DPE (datapoint element) called *counter* indicating the number of successive analysis's within the sequence. The counter is limited to 10 times, but currently not used. It is set to 1, which is also the default. If set to zero, the channel is rejected and added to the number of rejected channels. In this way channels which are definitely not working are put out of the sequence.

1.3.2 ConfDb

The ConfDb has the following attributes:

- Database: **atonr_conf**
- User: **ATLAS_CONF_MDT_W**
- Password: ****** (classified)

There is also a read-only account (ATLAS_MDT_DCS_R) for this database. The table used inside is hidden. Access is only possible by means of built-in functions (in order to protect the data and history). If needed the PVSS projects ATLMDTBAL[1-8] use the read-only account to setup their datapoints from scratch. The super-visor ATLMDTBAL9 however (PC9), uses the read-write account to synchronize the contents of the ConfDb with that of the datapoints. For this purpose a dedicated control manager takes care that for each channel the parameters are equalized. Hence any difference is copied from the DPE and written to the ConfDb. It runs every night. Items which are maintained are: the addresses of camera and light-source, the electronic settings of the camera (also known as the I^2C settings) and the parameters for the analysis.
As mentioned earlier the access is completely performed by built-in SQL functions. For instance to get the total list of channels of server *X* [1-8], the following command needs to be executed:

```
SELECT * FROM TABLE (atlas_conf_mdt.align_r.get_channel_for_camsrv(X))
```

A complete list of possible queries is given in appendix 4.1.

1.4 FSM

The barrel alignment system is fully integrated into the ATLAS-FSM control tree and it is part of the MDT leaf which itself resides under the MUON leaf. Alignment is in general passive, i.e. no commands are implemented and only its status is relevant. The FSM structure for the barrel alignment is shown in Figure 1.7. It is split into two main parts: barrel side A and side C, complying with the general partitioning scheme of ATLAS. Each of them has four DUs (Device Units), which are actually the running PVSS projects on the alignment PCs. PC[1-4] have only channels on side A, PC[5-8] only channels on side C. On PC9, the supervisory computer, a dedicated *Watchdog* PVSS-manager is running, determining the state and status of the DUs and CUs (Control Unit) above, based on for instance the average amount of analysis errors, the ability to write to the database, etc.

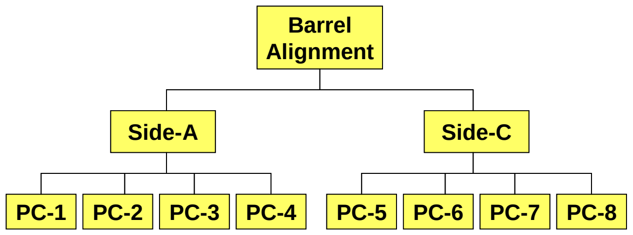


Figure 1.7: Barrel Alignment Hierarchy

2 User Manual

Three types of users can be distinguished:

1. DCS-experts: able to logon all PCs and do whatever is necessary.
2. Alignment-experts: able to log on PC9 only, adjusting the parameters of the channels.
3. Shifters and other users: not allowed to logon at any PC.

The entire readout system (PCs, PVSS projects, Rasdim servers, etc) is completely automated, i.e. at startup no human intervention is needed.

2.1 Production PCs

The productions PCs, **PCATLMDTBAL[1-8]**, have two major components:

1. **Rasdim**: The server handling the multiplexor, the frame-grabber and analysis.
2. **PVSS**: the client, sending commands to its Rasdim server, gathering the data and storing it into the database (CondDb).

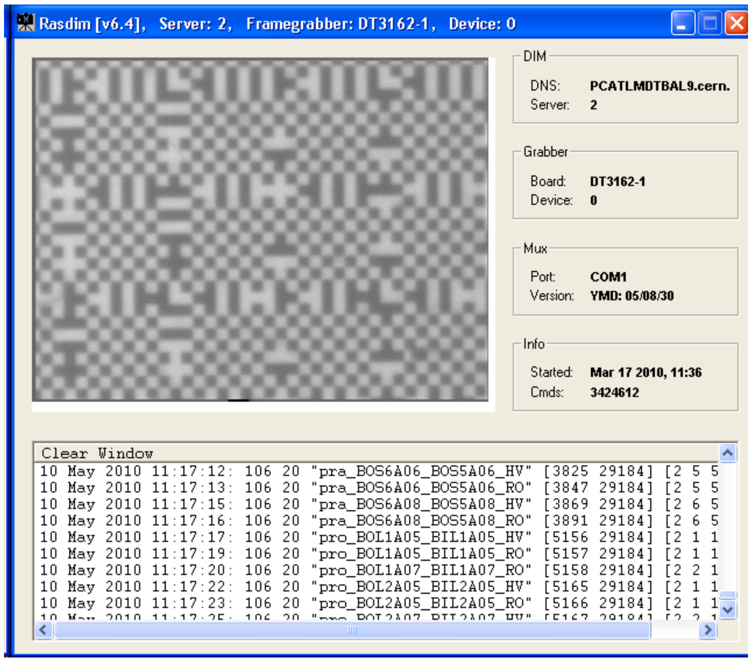
The PVSS projects are started as *service*, hence the managers run as user SYSTEM. All PVVSS-GUIs (PVSS00ui) are set to manual; hence no panel will appear at startup. If necessary the PVSS console may be opened and closed for debugging purposes only and by experts only. For more details see the implementation report.

The Rasdim server is started differently. The driver of the DT3162 frame-grabber is written in compliance with the Windows Driver Model (WDM) which implies the Rasdim server cannot be started remotely, e.g. not via remote desktop, nor as a service. In that case, the grabbing of the images fails due to a timeout (E_GRABBER=#3), because somehow the video signal is virtually not there. For this reason a local account is created: **balign**, which is logged on at startup after which the Rasdim server is started. This is implemented by two Visual Basic scripts. As a consequence `balign` is always logged on, leaving the console screen active. It is highly discouraged to logon remotely on one of the production PCs. If it is unavoidable, then logon using the local account and when finished do **NOT** log off or disconnect! Instead execute the following command line:

```
tscon /DEST:console 0
```

On the desktop a shortcut is available executing the command: Redirect To Console

Thus, please be careful logging on one of the production PCs. Modifying the parameters of certain channels is also possible on the super-visor PC9 (Chapter 2.4).
The only visible application running is the Rasdim server and an example is shown in Figure 2.1. It is allowed (and even recommended) to minimize the server. The Rasdim server is explained in more detail in the implementation section.



Rasdim [v6.4], Server: 2, Framegrabber: DT3162-1, Device: 0

DIM
DNS: PCATLMDTBAL9.cern.
Server: 2

Grabber
Board: DT3162-1
Device: 0

Mux
Port: COM1
Version: YMD: 05/08/30

Info
Started: Mar 17 2010, 11:36
Cmds: 3424612

```
Clear Window
10 May 2010 11:17:12: 106 20 "pra_BOS6A06_BOS5A06_HV" [3825 29184] [2 5 5
10 May 2010 11:17:13: 106 20 "pra_BOS6A06_BOS5A06_RO" [3847 29184] [2 5 5
10 May 2010 11:17:15: 106 20 "pra_BOS6A08_BOS5A08_HV" [3869 29184] [2 6 5
10 May 2010 11:17:16: 106 20 "pra_BOS6A08_BOS5A08_RO" [3891 29184] [2 6 5
10 May 2010 11:17:17: 106 20 "pro_BOL1A05_BIL1A05_HV" [5156 29184] [2 1 1
10 May 2010 11:17:19: 106 20 "pro_BOL1A05_BIL1A05_RO" [5157 29184] [2 1 1
10 May 2010 11:17:20: 106 20 "pro_BOL1A07_BIL1A07_RO" [5158 29184] [2 2 1
10 May 2010 11:17:22: 106 20 "pro_BOL2A05_BIL2A05_HV" [5165 29184] [2 1 1
10 May 2010 11:17:23: 106 20 "pro_BOL2A05_BIL2A05_RO" [5166 29184] [2 1 1
10 May 2010 11:17:25: 106 20 "pro_BOL2A07_BIL2A07_HV" [5167 29184] [2 2 1
```

Figure 2.1: Rasdim server

2.2 Super-visor PC

The super-visor PC, **PCATLMDTBAL9**, has also two major components:

1. **DNS:** the DIM name server: enables communication between the Rasdim servers and their clients (i.e. PVSS).
2. **PVSS:** holding the FSM part of the readout system.

The following remarks are valid for the super-visor PC:

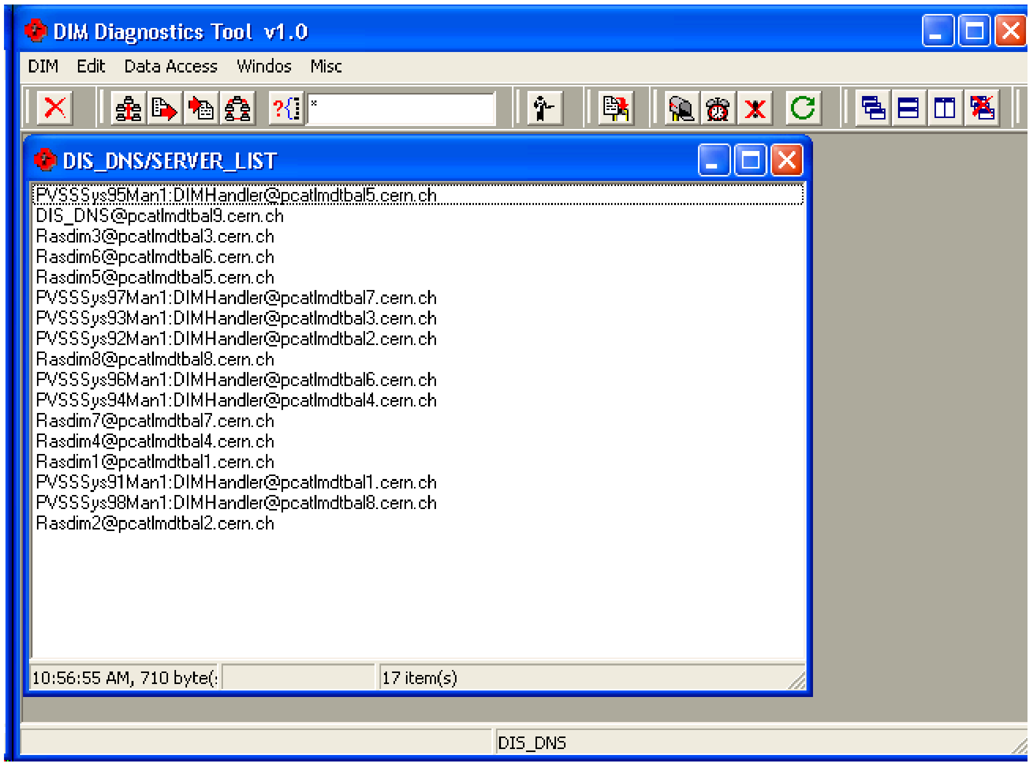
- PVSS and DNS are both started as *service*; hence they run under user SYSTEM.
- All PVSS-GUIs (PVSS00ui) are set to manual; hence no panel will appear at startup. More details of the PVSS project on PC9 can be found in the implementation report.
- Logging on and off, using your NICE account, on PC9 is allowed. Always log off, when you are finished!
- No visible applications are running at logon except for the *OLWatchGui* program. It takes care of a daily summary for statistical purposes. To be ignored!
- To manipulate the parameters of a particular channel, PC9 provide all means doing so. Chapter 2.4 describes this is more detail.

2.3 DIM

The Distributed Information Management (DIM) system is used for client/server communication. Its protocol requires a name server DNS running somewhere. The super-visor PC is used to hold the DNS. In order to make communication between Rasdim and PVSS possible, the following environment system variable has to be set on **all** PCs.

DIM_DNS_NODE	**pcatlmdtbal9.cern.ch**

DIM provides a tool, called *DID*, to check the existence of DNS and running applications. Executed on an arbitrary alignment PC, a typical output is shown in Figure 2.2.



```
PVSSSys95Man1:DIMHandler@pcatlmdtbal5.cern.ch
DIS_DNS@pcatlmdtbal9.cern.ch
Rasdim3@pcatlmdtbal3.cern.ch
Rasdim6@pcatlmdtbal6.cern.ch
Rasdim5@pcatlmdtbal5.cern.ch
PVSSSys97Man1:DIMHandler@pcatlmdtbal7.cern.ch
PVSSSys93Man1:DIMHandler@pcatlmdtbal3.cern.ch
PVSSSys92Man1:DIMHandler@pcatlmdtbal2.cern.ch
Rasdim8@pcatlmdtbal8.cern.ch
PVSSSys96Man1:DIMHandler@pcatlmdtbal6.cern.ch
PVSSSys94Man1:DIMHandler@pcatlmdtbal4.cern.ch
Rasdim7@pcatlmdtbal7.cern.ch
Rasdim4@pcatlmdtbal4.cern.ch
Rasdim1@pcatlmdtbal1.cern.ch
PVSSSys91Man1:DIMHandler@pcatlmdtbal1.cern.ch
PVSSSys98Man1:DIMHandler@pcatlmdtbal8.cern.ch
Rasdim2@pcatlmdtbal2.cern.ch

10:56:55 AM, 710 byte(:    17 item(s)
DIS_DNS
```

Figure 2.2: DID output

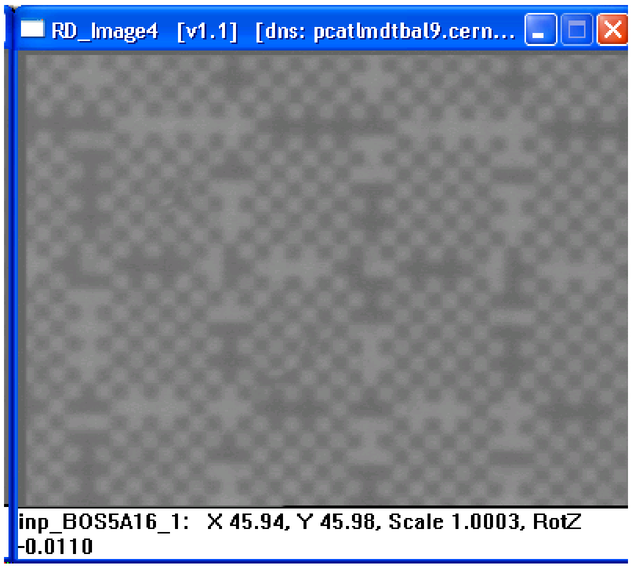
Inside the DIS_DNS/SERVER_LIST the following items should be present:

- `DIS_DNS@pcatlmdtbal9.cern.ch`: DNS the DIM name server
- `Rasdim`*`X`*`@pcatlmdtbal`*`X`*`.cern.ch`: Rasdim server *X* [1-8]

2.4 Channel Manipulation

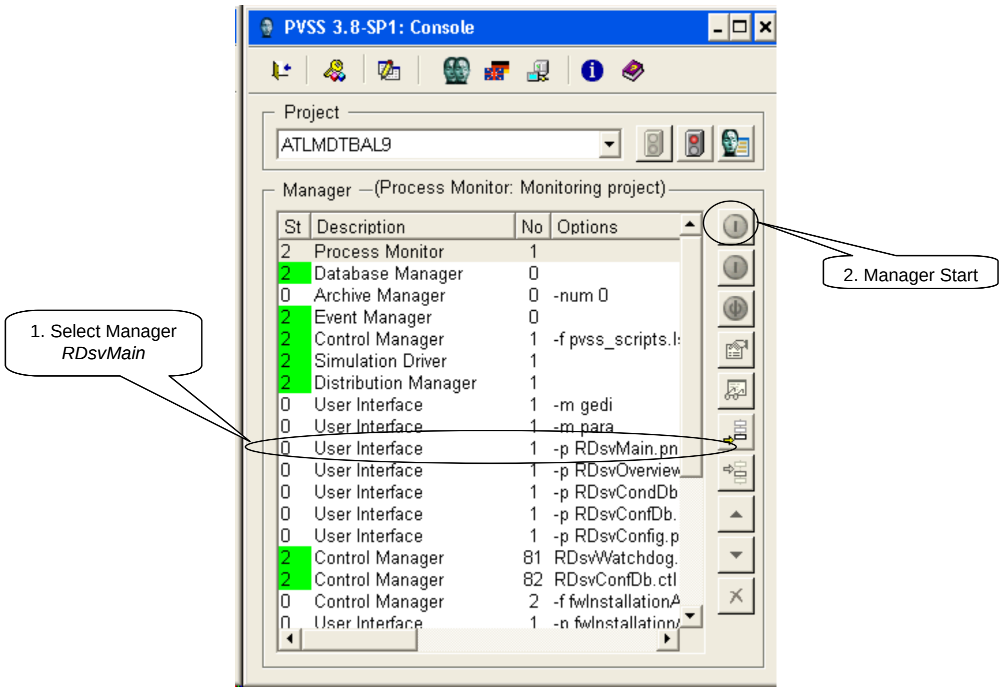
Whenever an alignment expert wants to modify the parameters of a particular channel, the following sequence has to be carried out. On beforehand the expert needs to know to which Server/PC the channel belongs.

1. Logon the super-visor PC9, using your NICE account.
2. Select and start the image-viewer of the PC the channel belongs to. For each server there is a separate executable. They reside in `C:\Balign\bin\image`. A shortcut on the desktop for each server might be convenient. As an example Figure 2.3 shows the viewer of server #4. During normal operation the viewer shows the images of the channels at a rate of 1 Hz.
3. Startup the PVSS console. Again a shortcut on the desktop might be convenient. After startup the console appears as shown in Figure 2.4.



```
inp_BOS5A16_1:  X 45.94, Y 45.98, Scale 1.0003, RotZ
-0.0110
```

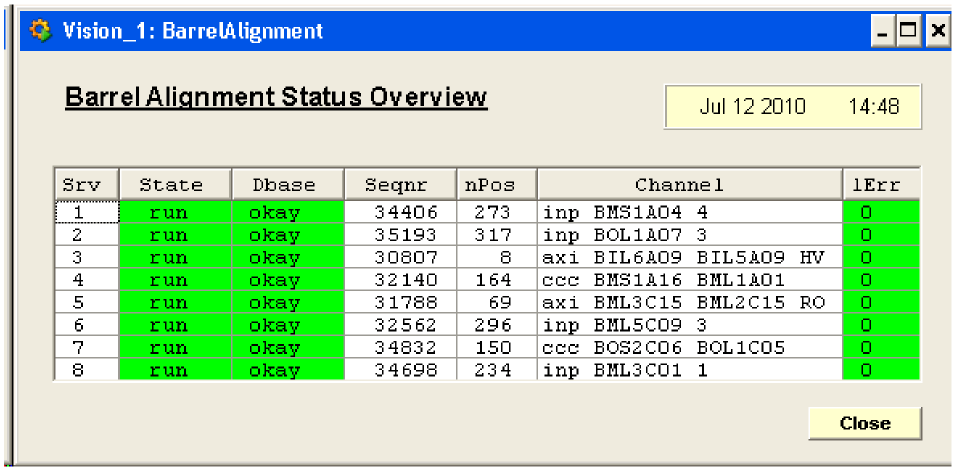
Figure 2.3: Image viewer of Server #4


PVSS 3.8-SP1: Console
Project
ATLMDTBAL9
Manager
(Process Monitor: Monitoring project)
1. Select Manager RDsvMain
2. Manager Start

St	Description	No	Options
2	Process Monitor	1	
2	Database Manager	0	
0	Archive Manager	0	-num 0
2	Event Manager	0	
2	Control Manager	1	-f pvss_scripts.l
2	Simulation Driver	1	
2	Distribution Manager	1	
0	User Interface	1	-m gedi
0	User Interface	1	-m para
0	User Interface	1	-p RDsvMain.pn
0	User Interface	1	-p RDsvOverview
0	User Interface	1	-p RDsvCondDb
0	User Interface	1	-p RDsvConfDb.
0	User Interface	1	-p RDsvConfig.p
2	Control Manager	81	RDsvWatchdog.
2	Control Manager	82	RDsvConfDb.ctl
0	Control Manager	2	-f fwInstallationA
0	User Interface	1	-p fwInstallation/

Figure 2.4: Start of RDsvMain panel

4. Select the user interface manager: **`RDsvMain`**. Start it by pressing the green push-button (the Start Manager button turns green when you select a manager). After a moment a panel will appear as shown in Figure 2.5. It is a summary and status overview of the eight servers.
5. Double click on the Server/PC the channel belongs to.



Srv	State	Dbase	Seqnr	nPos	Channel	lErr
1	run	okay	34406	273	inp BMS1A04 4	0
2	run	okay	35193	317	inp BOL1A07 3	0
3	run	okay	30807	8	axi BIL6A09 BIL5A09 HV	0
4	run	okay	32140	164	ccc BMS1A16 BML1A01	0
5	run	okay	31788	69	axi BML3C15 BML2C15 RO	0
6	run	okay	32562	296	inp BML5C09 3	0
7	run	okay	34832	150	ccc BOS2C06 BOL1C05	0
8	run	okay	34698	234	inp BML3C01 1	0

Figure 2.5: RDsvMain panel

6. The main server panel, shown in Figure 2.6, will appear. As an example server #1 is used. The special sequence buttons (`Control` and `Modify`) should not be used, they are for experts only. The number of channels of server #1 is 707 and the state is ON (indicating the sequence is running). In order to specify a specific channel, press the `Edit Channels` button.

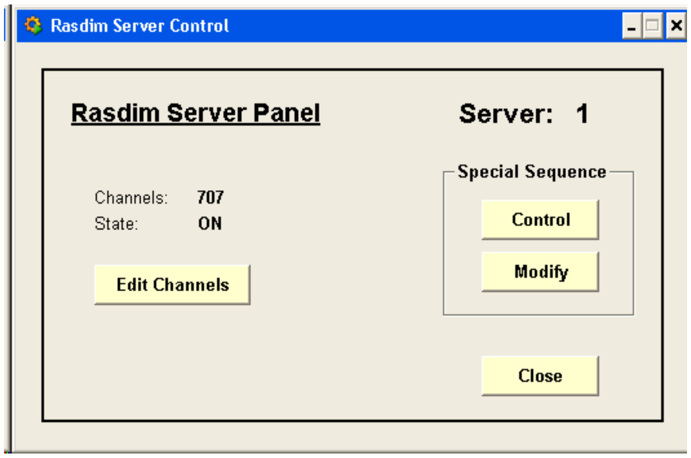
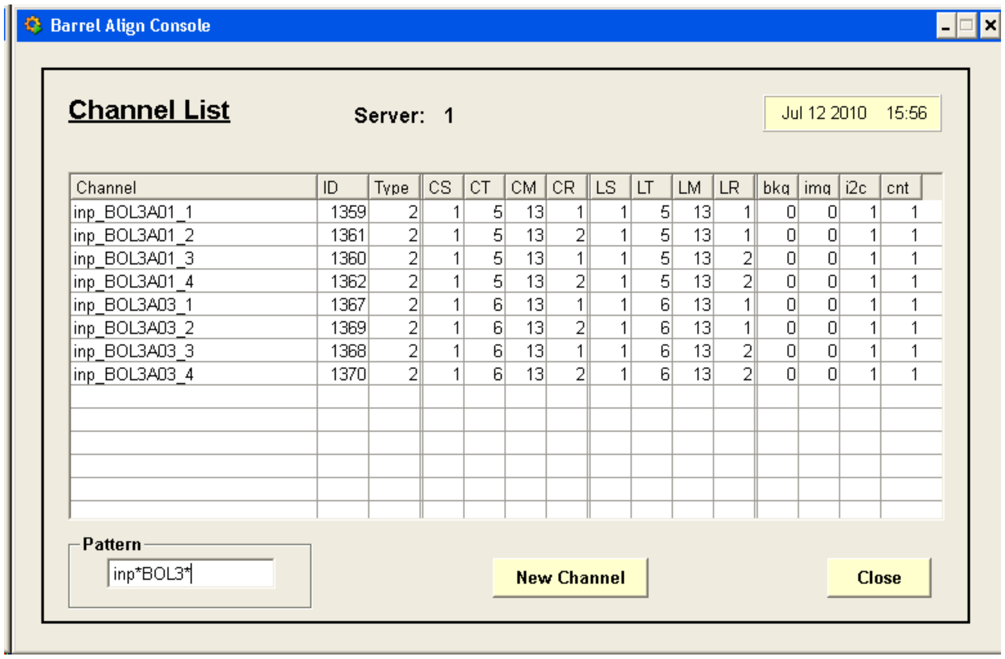


Figure 2.6: Main Server panel

7. The channel list is displayed in Figure 2.7. The `New Channel` button is for experts only. Initially the list is empty. By means of the `Pattern` selection, in which any regular expression is possible, a subset of the channels is listed. Make sure the input focus is set to this widget and press `Enter` to evaluate the pattern. The default is all '*', but it will take some time to fill the list with 707 channels. Figure 2.7 shows the list of channels as a result of the pattern: `inp*BOL3*.` Pressing a column header will reverse the order of the list, belonging to that header. The columns which are displayed are a subset (but an important subset) of the parameters. From left to right:
   - *ID*: a unique number of the channel.
   - *Type*: analysis (Rasnik #0, Spot #1 or Foam #2)
   - *CS*, *CT*, *CM*, *CR*: Camera address for the server, TopMux, MasterMux and RasMux respectively (NB: The *CS* number has to be equal to the server number on top of the panel)
   - *LS*, *LT*, *LM*, *LR*: Led address: for the server, TopMux, MasterMux and RasMux respectively.
   - *Bkg*: background subtracting; in general set to 0, meaning not used.
   - *Img*: number of images taken in order to get an average image. In general not used; set to 0.
   - I^2C: at least one of the parameters is used (the majority of channels uses I^2C).

- *Cnt*: the number of successive analysis within a sequence. Normally set to 1. If set to 0, the channel is taken out of the sequence.

The *Bkg*, *Img* and *Cnt* items seem a little exotic, but could have a severe influence on the sequence. They are shown in this panel, in order to have a quick overview of them. The server number *CS* has to match with the server number on top of the panel. After abandoning the mixed-channel concept, the server number of the Led (*LS*) has to match as well, i.e. *CS*=*LS*=*Server*.

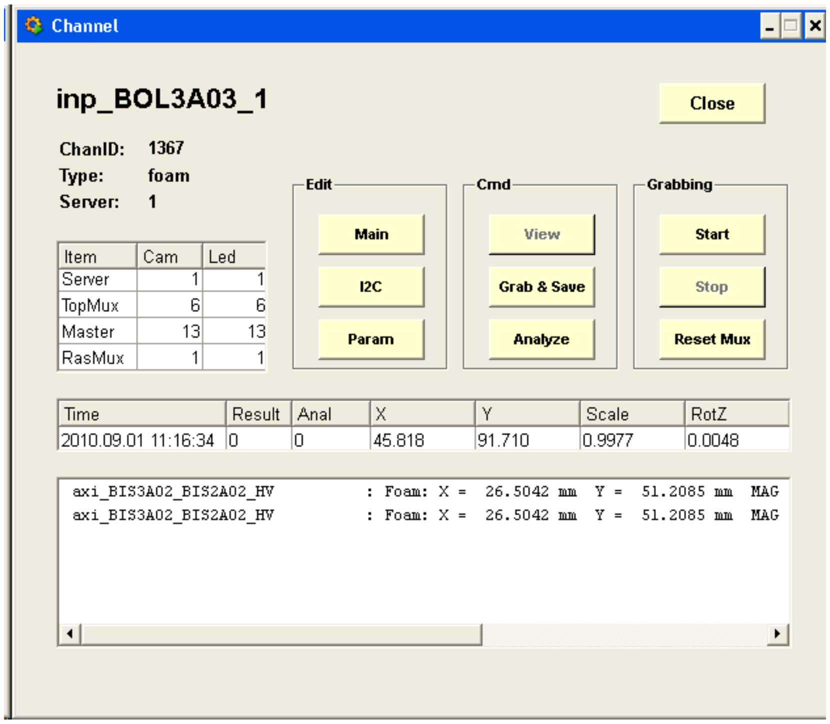


Channel List Server: 1 Jul 12 2010 15:56

Channel	ID	Type	CS	CT	CM	CR	LS	LT	LM	LR	bkg	img	i2c	cnt
inp_BOL3A01_1	1359	2	1	5	13	1	1	5	13	1	0	0	1	1
inp_BOL3A01_2	1361	2	1	5	13	2	1	5	13	1	0	0	1	1
inp_BOL3A01_3	1360	2	1	5	13	1	1	5	13	2	0	0	1	1
inp_BOL3A01_4	1362	2	1	5	13	2	1	5	13	2	0	0	1	1
inp_BOL3A03_1	1367	2	1	6	13	1	1	6	13	1	0	0	1	1
inp_BOL3A03_2	1369	2	1	6	13	2	1	6	13	1	0	0	1	1
inp_BOL3A03_3	1368	2	1	6	13	1	1	6	13	2	0	0	1	1
inp_BOL3A03_4	1370	2	1	6	13	2	1	6	13	2	0	0	1	1

Figure 2.7: Channel list

8. Double click on the channel to be checked. As a result a panel will appear as shown in Figure 2.8. **NB**: *Be aware that from now on the sequence is stopped. The state as shown in Figure 2.5 and Figure 2.6 will go to OFF. It will result into* ***ALARMS*** *and* ***ERRORS*** *in the ATLAS Control Room (they will propagate to the central DCS desk). Make sure you have notified the shifter and shift leader.* As soon as you exit/close/quit this panel, within seconds the sequence will be resumed. On the left upper side of the panel the main parameters of the channel are shown: its name, analysis type, the server and the addresses of the Camera and Led. The result of the last analysis is shown in the middle table. At the bottom a message window shows some information (if any) on the actions taken. The 'real' action/commands are grouped into three sets: *Edit*, *Cmd* and *Grabbing*.
9. *Grabbing*: to be used whenever there is serious doubt on the optical connection to the channel. Pressing the `Start` button triggers the Rasdim server to take images as fast as possible. A picture, as shown in Figure 2.3, should be seen. The `Stop` button terminates the grabbing. Another possibility is to reset the TopMux (by means of the `Reset Mux` button).
10. *Cmd*: Basic functionality of the Rasdim server. The `View` button is permanently disabled. It was used to set the camera into continuous acquire mode (real-time video), but it caused sometimes bad behavior of the Rasdim server. The `Grab & Save` button performs a single acquire, grabbing an image and saves it as a TIF file. Finally the `Analyze` button does the same, but instead of saving the image, it analyses it. The result will appear in the middle table.



inp_BOL3A03_1

ChanID:	1367
Type:	foam
Server:	1

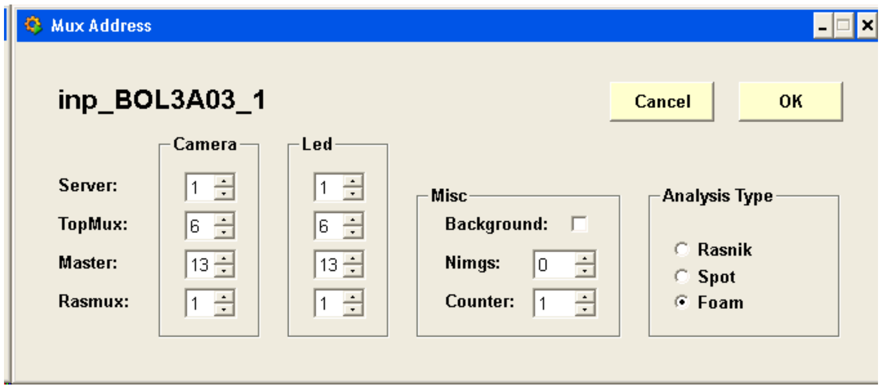
Item	Cam	Led
Server	1	1
TopMux	6	6
Master	13	13
RasMux	1	1

Time	Result	Anal	X	Y	Scale	RotZ
2010.09.01 11:16:34	0	0	45.818	91.710	0.9977	0.0048

```
axi_BIS3A02_BIS2A02_HV          : Foam: X =  26.5042 mm  Y =  51.2085 mm  MAG
axi_BIS3A02_BIS2A02_HV          : Foam: X =  26.5042 mm  Y =  51.2085 mm  MAG
```

Figure 2.8: Channel panel

11. *Edit*: The parameters of a channel are separated into three groups: *Main*, *I^{2}C* and *Params*. Be careful changing any parameter.
12. *Main*: This panel, entitled confusingly `Mux Address`, is shown in Figure 2.9. It contains the most sensitive parameters, in particular the Camera and Led address. Also the `Analysis Type` should be handled with great prudence. Only the 'real' experts should consult this panel.



inp_BOL3A03_1

	Camera	Led
Server:	1	1
TopMux:	6	6
Master:	13	13
Rasmux:	1	1

Misc

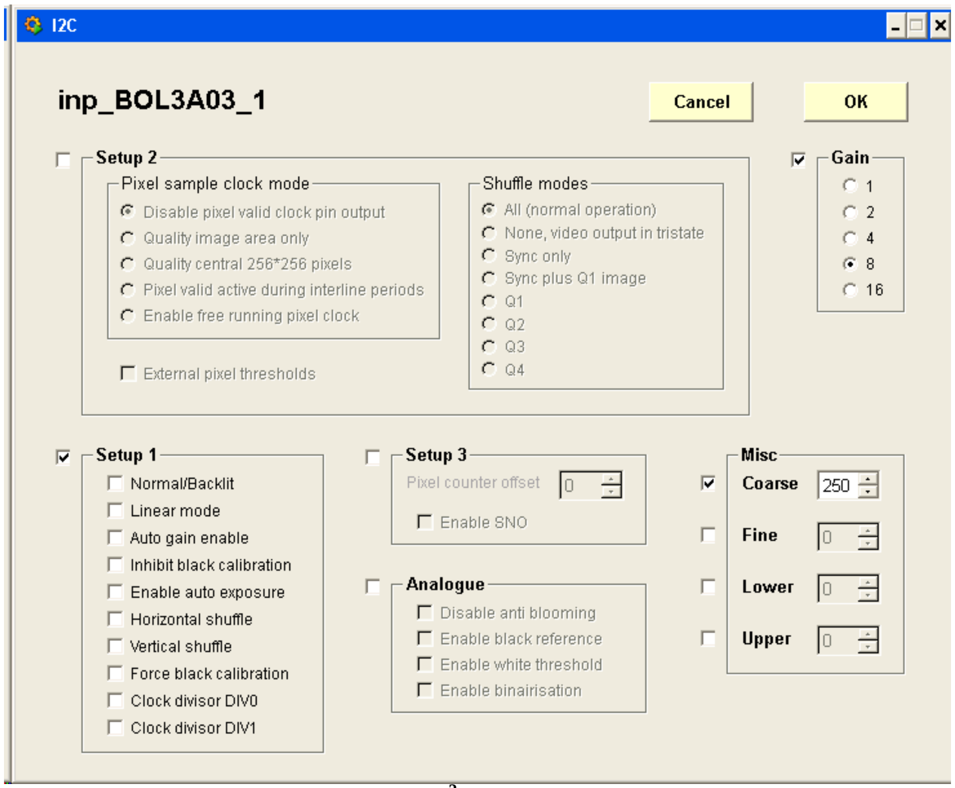
Background:	☐
Nimgs:	0
Counter:	1

Analysis Type

- ○ Rasnik
- ○ Spot
- ● Foam

Figure 2.9: Main Parameter panel

13. *I^{2}C*: The camera contains a lot of optical settings, managed by the I^{2}C electronics bus. Figure 2.10 shows the possible settings.
14. *Param*: This subject denotes the parameters of the analysis and is therefore dependent on the chosen type. Figure 2.11 shows the parameters of analysis type Foam, Figure 2.12 of type Spot and for completeness Figure 2.13 of type Rasnik.



inp_BOL3A03_1

Setup 2
- Pixel sample clock mode: [✓ Disable pixel valid clock pin output] / Quality image area only / Quality central 256*256 pixels / Pixel valid active during interline periods / Enable free running pixel clock
- External pixel thresholds
- Shuffle modes: [✓ All (normal operation)] / None, video output in tristate / Sync only / Sync plus Q1 image / Q1 / Q2 / Q3 / Q4

[✓ Gain]: 1 / 2 / 4 / [✓ 8] / 16

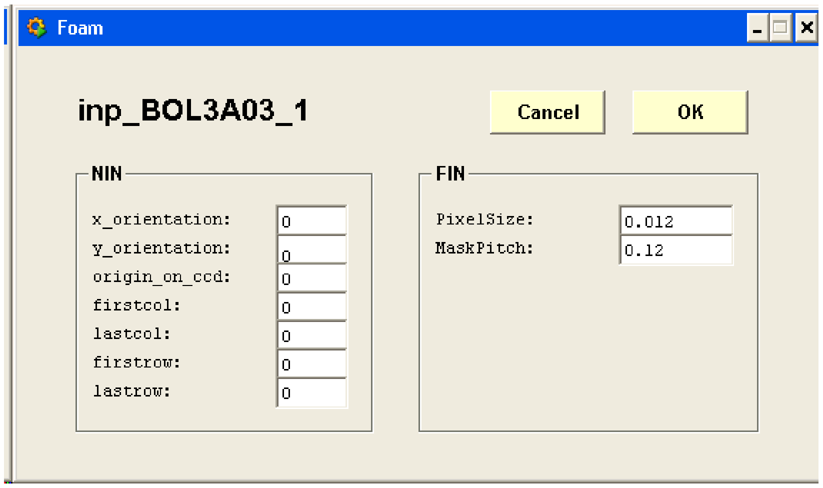
[✓ Setup 1]
- Normal/Backlit
- Linear mode
- Auto gain enable
- Inhibit black calibration
- Enable auto exposure
- Horizontal shuffle
- Vertical shuffle
- Force black calibration
- Clock divisor DIV0
- Clock divisor DIV1

Setup 3
- Pixel counter offset: 0
- Enable SNO

Analogue
- Disable anti blooming
- Enable black reference
- Enable white threshold
- Enable binairisation

Misc
- [✓ Coarse]: 250
- Fine: 0
- Lower: 0
- Upper: 0

Figure 2.10: I^{2}C camera settings

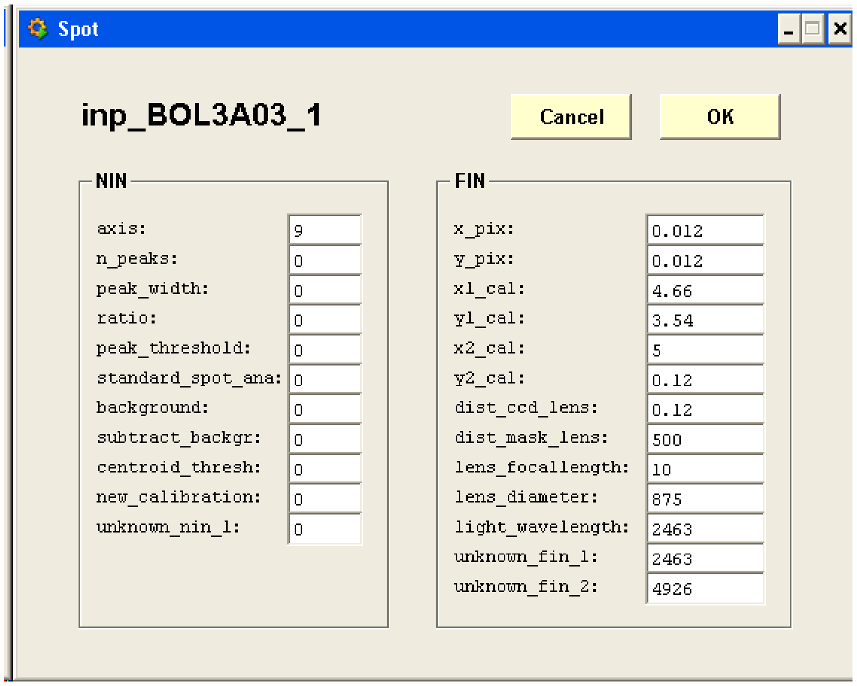


inp_BOL3A03_1

NIN	
x_orientation:	0
y_orientation:	0
origin_on_ccd:	0
firstcol:	0
lastcol:	0
firstrow:	0
lastrow:	0

FIN	
PixelSize:	0.012
MaskPitch:	0.12

Figure 2.11: Foam analysis parameters



inp_BOL3A03_1

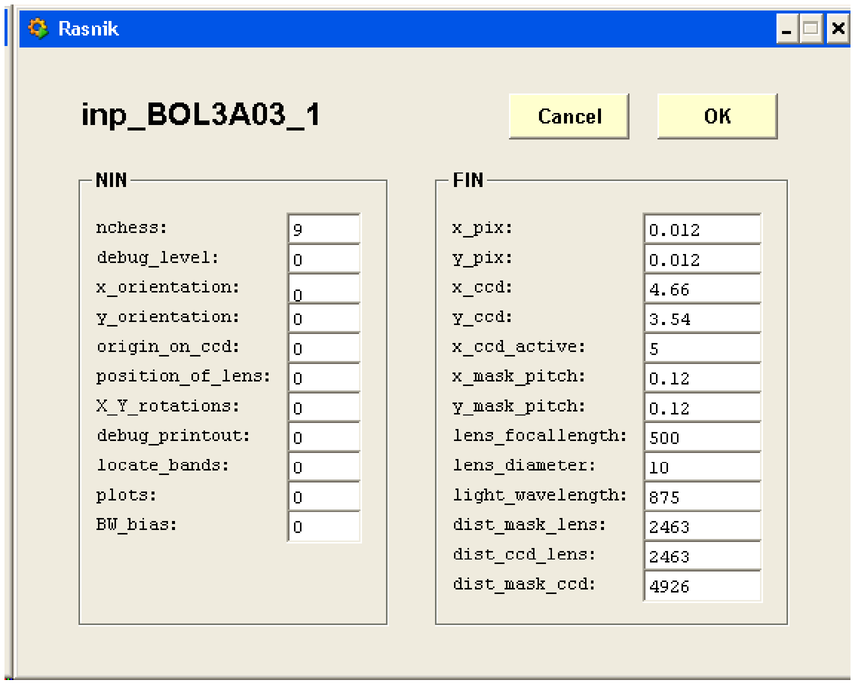
NIN

axis:	9
n_peaks:	0
peak_width:	0
ratio:	0
peak_threshold:	0
standard_spot_ana:	0
background:	0
subtract_backgr:	0
centroid_thresh:	0
new_calibration:	0
unknown_nin_1:	0

FIN

x_pix:	0.012
y_pix:	0.012
x1_cal:	4.66
y1_cal:	3.54
x2_cal:	5
y2_cal:	0.12
dist_ccd_lens:	0.12
dist_mask_lens:	500
lens_focallength:	10
lens_diameter:	875
light_wavelength:	2463
unknown_fin_1:	2463
unknown_fin_2:	4926

Figure 2.12: Spot analysis parameters



inp_BOL3A03_1

NIN

nchess:	9
debug_level:	0
x_orientation:	0
y_orientation:	0
origin_on_ccd:	0
position_of_lens:	0
X_Y_rotations:	0
debug_printout:	0
locate_bands:	0
plots:	0
BW_bias:	0

FIN

x_pix:	0.012
y_pix:	0.012
x_ccd:	4.66
y_ccd:	3.54
x_ccd_active:	5
x_mask_pitch:	0.12
y_mask_pitch:	0.12
lens_focallength:	500
lens_diameter:	10
light_wavelength:	875
dist_mask_lens:	2463
dist_ccd_lens:	2463
dist_mask_ccd:	4926

Figure 2.13: Rasnik analysis parameters

2.5 Using Spare PC10

Spare PC10 (pcatlmdtbal10) is meant to take over the functionality of a production PC [1-8] in case of a malfunction. This chapter describes the steps necessary to do so. Remember that up to now, it never has been performed or tested yet. Hence, flaws and omissions inside the procedure may be present. The following prerequisites, for the person taking care of this action, are assumed to be present and valid:

- Access rights to enter USA15.
- Knowledge of PVSS: i.e. knowing how to create/start a PVSS project and add/start a manager.
- Knowledge of the JCOP framework: knowing how to utilize and add components.
- The latest version of PVSS should be installed (3.8 SP1 or higher), together with a valid (indefinite) license.
- The latest version of Rasdim (6.4 or higher) should be present as well, including the latest versions of the different analysis modules.
- It is assumed PC10 is maintained well, i.e. the latest patches and updates are installed

The next steps have to be carried out in case PC*x* ($1 \leq x \leq 8$) fails and has to be taken over by PC10. It means that PC10 has to be connected to the hardware of server *x* and should run the same PVSS project.

1. Disconnect the frame-grabber and serial-port (COM1) cable from PC*x* and connect them to PC10.
2. Reboot PC10 and logon, using the local account **balign**.
3. Check the connectivity by means of the *mmm* and *fgTest* application. Both executables are found as shortcut on the desktop. Start with the mmm application and try to connect both the camera and light-source to TopMux #1, MasterMux #1 and RasMux #1. On all servers there should be a existing channel behind this base address. The fgTest application should now show an image. If not, check the cables and/or reboot. On success, please continue, but do not forget to exit the two applications!
4. The panels and scripts of the PVSS projects are kept and maintained at one place and made available offline. Therefore it is necessary to map the following network drive (if not done already): `L:\\atlas-storage-set-dcs`
5. Create a *distributed* PVSS project (using the PVSS project Administration utility) with the following settings (take the default for the other settings):
   - Project name: `ATLMDTBAL`*`x`*
   - Path: `C:\pvss`
   - System number: `90 + `*`x`*
   - System name: `ATLMDTBAL`*`x`*
6. Make sure that the folders `C:\pvss`, as well as the folder `C:\pvss\ATLMDTBAL`*`x`*, are *shared*.
7. Install the JCOP framework following the instructions found in:
   https://edms.cern.ch/file/1055675/4.0.0/fwReadme.txt and install the following components:
   - `fwAtlas`
   - `fwAtlasInstallation`
   - `fwDIM`

   The result has to be placed into the component folder:
   `C:\pvss\fwComponents_ATLMDTBAL`*`x`*
8. The project's configuration file: verify that the general part of the `config` file has a layout as shown in Script 2.1 and that in both the `[ctrl]` as `[ui]` part the line "`CtrlDLL = CtrlRDBAccess`" is included. If not, adjust it by hand, respecting the order by which the `proj_paths` are placed.
9. Besides the system configuration file of PVSS, there is also a 'user' configuration file for the project itself. It is called **`RDconfig.ini`** and resides in the `config` area as well. Take a copy from another project and place it inside this area.

```
[general]
pvss_path = "C:/ETM/PVSS2/3.8"
proj_path = "\\atlas-storage-det-dcs\DCS\fwInstallation\fwInstallation"
proj_path = "\\atlas-storage-det-dcs\DCS\PVSSProjects\ATLAS_DCS_MDT"
proj_path = "\\atlas-storage-det-dcs\DCS\PVSSProjects\ATLAS_DCS_MDT\ATLMDTBAL"
proj_path = "\\atlas-storage-det-dcs\DCS\PVSSProjects\ATLAS_DCS_MDT\SW_EXTRAS"
proj_path = "C:\pvss\fwComponents_ATLMDTBALx"
proj_path = "C:/pvss/ATLMDTBALx"

[ui]
CtrlDLL = "CtrlRDBAccess"

[ctrl]
CtrlDLL = "CtrlRDBAccess"
```

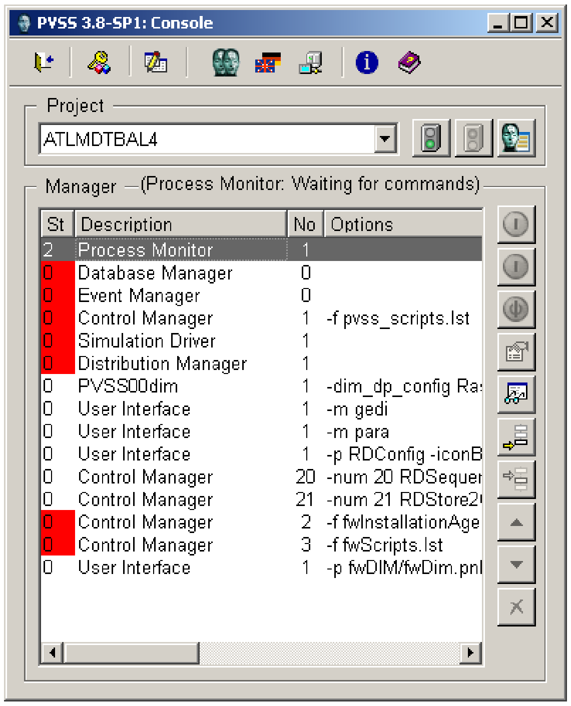
Script 2.1: PVSS config file

10. From now on PC/Server #4 is taken as example for the remainder of the procedure: thus ***`x = 4`***.
11. Open the PVSS console and select project ATLMDTBAL4. Delete the six archive managers and add the following managers (Table 2.1):

Manager	Arguments
`PVSS00dim`	`-dim dp config Rasdim`
`User Interface`	`-m para`
`User Interface`	`-p RDConfig –iconBar –menuBar`
`PVSSctrl`	`-num 20 RDSequence.ctl`
`PVSSctrl`	`-num 21 RDStore2ConfDb.ctl`

Table 2.1: Additional managers

Set the start mode to **`manual`**, including the already present User Interface manager [`-m gedi`]. The console should now look as in Figure 2.14:



```
Project: ATLMDTBAL4
Manager (Process Monitor: Waiting for commands)

St  Description          No  Options
2   Process Monitor       1
0   Database Manager      0
0   Event Manager         0
0   Control Manager       1  -f pvss_scripts.lst
0   Simulation Driver     1
0   Distribution Manager  1
0   PVSS00dim             1  -dim_dp_config Ra:
0   User Interface        1  -m gedi
0   User Interface        1  -m para
0   User Interface        1  -p RDConfig -iconB
0   Control Manager      20  -num 20 RDSequer
0   Control Manager      21  -num 21 RDStore2
0   Control Manager       2  -f fwInstallationAge
0   Control Manager       3  -f fwScripts.lst
0   User Interface        1  -p fwDIM/fwDim.pnl
```

Figure 2.14: PVSS console

12. Start the project and start manually the user interface manager `RDConfig`. Its panel should look as in Figure 2.15.

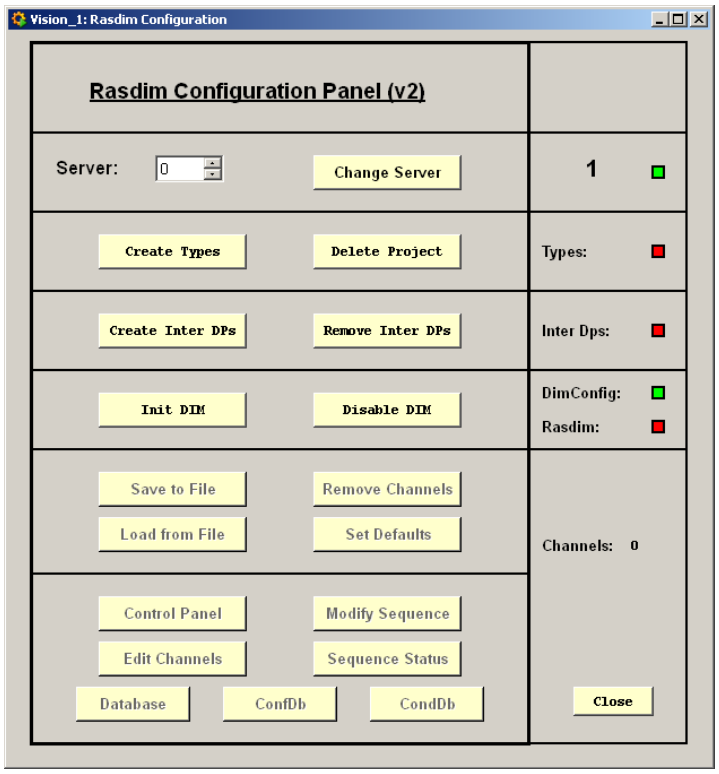


Figure 2.15: Initial configuration panel

13. Two of the tiny square status indicators are initially green (okay). The `DimConfig` one indicates that the DIM component of the framework is installed correctly. If not, go back to step 7 of the procedure. By default the server number is set to #1. In our case it should be set to #4 and it has to be the first action to perform, using the spin widget and pressing the `Change Server` button.
14. To finish the basic setup press the following buttons -one by one- in the following order (wait until the corresponding indicator turns green before you continue):
    - `Create Types`
    - `Create Inter DPs`
    - `Init DIM`
15. The PVSS-Rasdim project is now initialized. The datapoint types are defined, the internal communication datapoints are created and the connection to DIM is established. All status indicators should be green, and the buttons of the lower part enabled, as shown in Figure 2.16.
16. The only part missing are the channel datapoints. Select the `ConfDb` button at the lower part of the configuration panel. A panel will appear as shown in Figure 2.17.

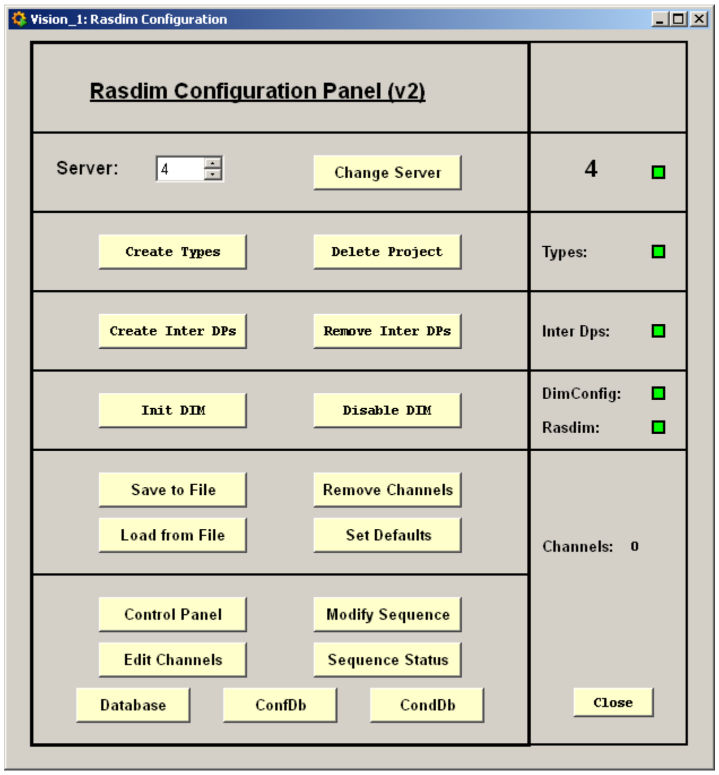
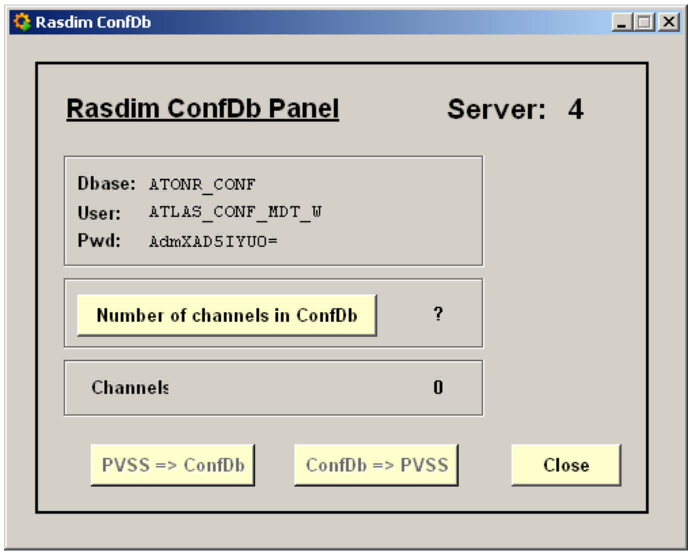


Figure 2.16: Configuration panel (2)



Dbase: ATONR_CONF
User: ATLAS_CONF_MDT_W
Pwd: AdmXAD5IYUO=

Channels 0

Figure 2.17: ConfDb panel

17. Press the button ‘`Number of channels in ConfDb`’. It will determine how many channels exist for server #4. After a couple of seconds the number 759 will appear and also the two buttons, `PVSS ==> ConfDb` and `ConfDb ==> PVSS`, are enabled. Press the `ConfDb ==> PVSS` button (be careful, not the other one!) in order to get the channels, including their parameters, into PVSS as datapoints. During this process the number of channels will increase. When finished close this panel and also the Configuration (parent)panel.

18. Stop the project and set the start mode of the following managers to **`always`**:
    - `PVSS00dim          –dim_dp_config Rasdim`
    - `Control Manager    -num 20 RDSequence.ctl`
    - `Control Manager    –num 21 RDStore2ConfDb.ctl`
19. At this stage the PVSS project is ready to perform its task. However, before PVSS can be started, the Rasdim server has to run first. There should be a shortcut on the desktop, but before executing, adjust the corresponding initialization file first. It resides in the same folder as Rasdim itself, which is (depending on the version of Rasdim): `C:\Balign\bin\Rasdim64`. Make sure that the entry `LogicServer` is set to #4. Now Rasdim can be executed. Figure 2.1 shows an example of the layout of the server. Verify that the `Server` is set to #4 and the `DNS` field equals: `PCATLMDTBAL9.cern.ch`.
20. Start the PVSS project. After a while all managers, with start mode set to **`always`**, will run. The `PVSS00dim` manager takes care of the communication and the `RDSequence` manager loops through all its channels and takes care of storing the results into the database.
21. Restart the PVSS project on the super-visor PC9, so the complete status of the barrel alignment can be determined again by (re-)connecting PC9 with `ATLMDTBAL`*`x`*.
22. Leave PC10 as it is. Do not logoff or disconnect! The procedure is finished now, but not entirely. PVSS does not run as *service* and when rebooting, the complete system (Rasdim and PVSS) will not be (re)started. The implementation chapter contains a section how to modify an alignment PC in order to restart it without human interaction.

NB: Remember that the procedure described above, has never been tested. Omissions and obscurities may be present.

3 Operational Aspects

During normal operation, i.e. the production PCs are up and running and the super-visor PC checks the states, a small set of PVSS managers takes care of the functionality of the alignment readout system. The following chapters describe the operational aspects and the managers in more detail.

3.1 RDSequence

The RDSequence manager, running on all production PCs with logical number #20, is the beating heart of the system which sets everything into motion. It makes a list of all channels, and handles them one after another in an infinite loop. After completion the list is handled again. A complete loop through the list is called a **sequence**. The behavior of the manager is completely determined by a set of datapoints. The important ones are:

- **state** (bool) the *run*-state of the sequence
- **seqnr** (unsigned) the current sequence number
- **npos** (unsigned) the index pointer inside the channel list

Figure 3.1 shows a simplified flowchart of the RDSequence manager.

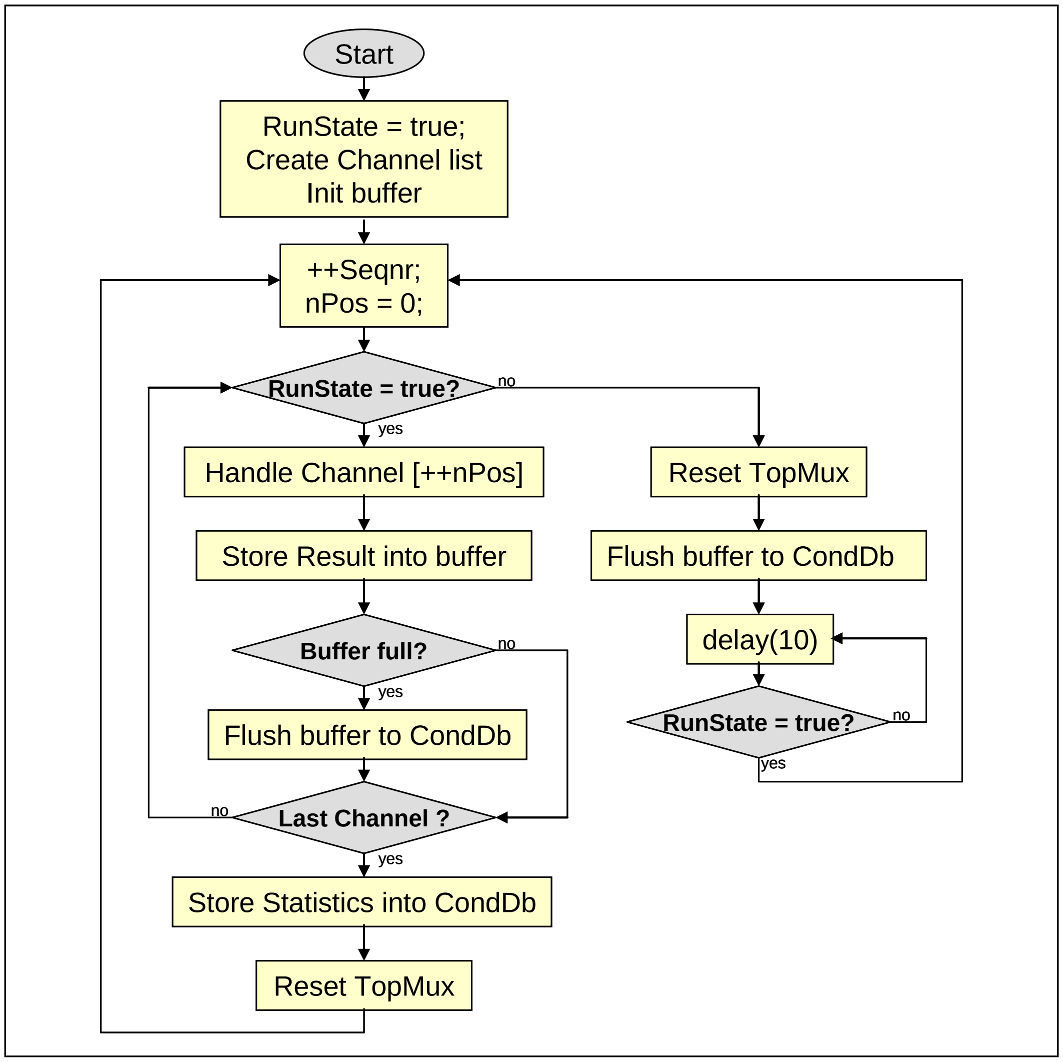
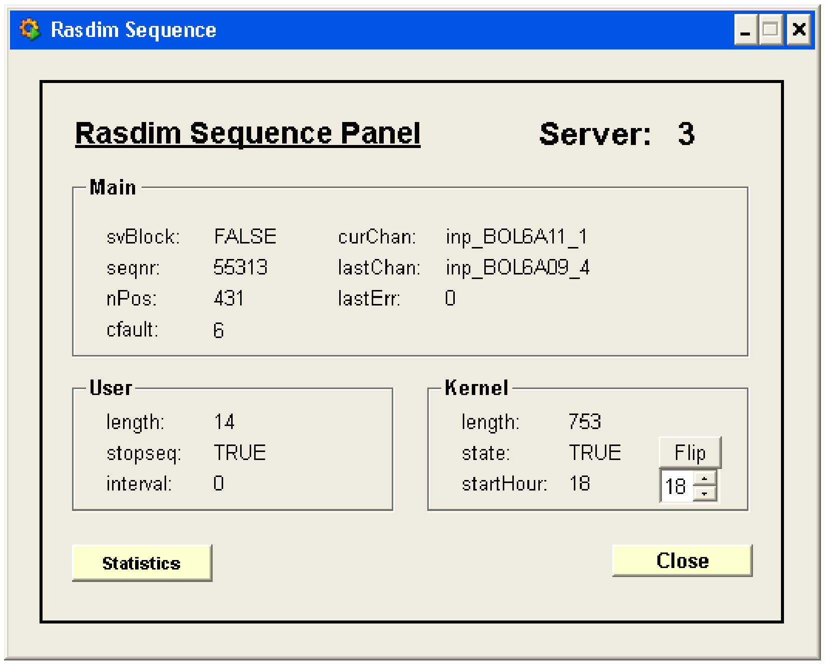


Figure 3.1: RDSequence flowchart

The RDSequence control manager uses a buffer for temporary storage of the results. The current size is 100 (i.e. the results of 100 analyses using a PVSS `dyn_dyn_mixed` type variable). Its primary use is to minimize the access load to the database. When full, a *handle* to the database is created and by means of a single bulk operation and bound variables the buffer is written to the database and emptied (see also Chapter 1.3.1 and Table 1.5 in particular). The status (i.e. success) of the storage is maintained in a separate datapoint of type *bool*. Another datapoint, named `storage` and not included in the flowchart, is used to skip the storage to the database. It was used in the past during maintenance when the data was not relevant. Nowadays its value is always set to *true*.
The “`Handle Channel`” box in the flowchart is described in more detail in the Implementation Notes chapter. Several statistic variables are maintained inside this module, of which the `errFraction` is the most important one. It holds the fraction (percentage) of errors of the last 500 analyses. During a sequence the number of errors (and types of errors) is kept up. At the end of a sequence these data is gathered and stored into the CondDb (see also Table 1.7). The total number of errors per day is also maintained.
The `state` datapoint denotes the *run*-state of the sequence. It could be set *false* from within inside (certain panels could do so) or more likely from outside (the super-visor in particular). Especially when a user wants to modify some channel parameters, the state variable is set to *false*. The first action to do is to switch off the TopMux, preventing a possible long time powering of the last selected camera and light source. During the setup of the barrel alignment system, this happened almost every day. A safety mechanism, not shown in the flowchart, is built-in in order to avoid endless suspension of the sequence. A datapoint, called `startHour`, contains the hour in which the sequence will be resumed. By default its value is set to 18, which means that the sequence restarts at 06:00 pm. It is allowed however to set the value to for instance 25. In that case the mechanism will not work and the sequence is stopped during the lifetime of RDSequence manager. Restarting the manager itself will always resume the sequence. The first action it does is setting its *run*-state to *true*.
The first sequence after midnight is the so-called *Saclay*-sequence. This sequence is a normal sequence, but with the following additions: the results are also stored into a dedicated file inside the data folder of the PVSS project and the images grabbed are saved. During normal operation the images grabbed are not saved, in order to limit the disk storage (see also Appendix 4.3 for the location of the saved images). It also resets the daily statistic parameters/datapoints.
The sequence panel in Figure 3.2 shows the relevant parameters and actual state of the RDSequence control manager. This panel is used for diagnostics and is only shown as illustration.



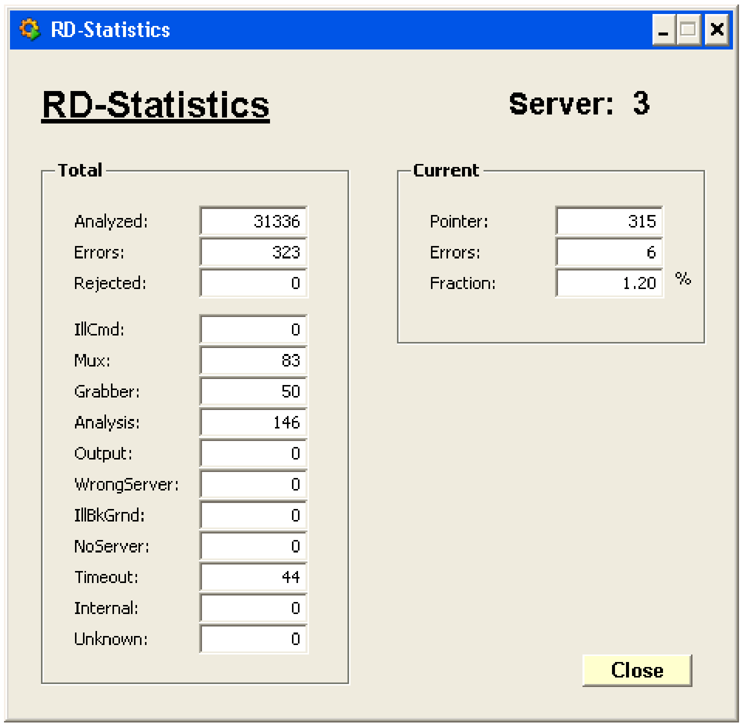
svBlock: FALSE
seqnr: 55313
nPos: 431
cfault: 6
curChan: inp_BOL6A11_1
lastChan: inp_BOL6A09_4
lastErr: 0

length: 14
stopseq: TRUE
interval: 0

length: 753
state: TRUE
startHour: 18
18

Figure 3.2: RDSequence panel

The `Kernel` group inside the RDSequence panel contains the total number (length) of channels, the *run*-state and `startHour`. The `Flip` button toggles the *run*-state. The `User` group shows some details of a user defined sequence. After the completion of the readout system, it became obsolete. The `Main` group shows some parameters of the current sequence. The `svBlock` parameter is a legacy of the mixed-channels concept and has to be *false*. The `cfault` parameter denotes the number of errors during the current sequence. Pressing the Statistic button shows a panel as shown in Figure 3.3.



Server: 3

Total

Analyzed:	31336
Errors:	323
Rejected:	0
IllCmd:	0
Mux:	83
Grabber:	50
Analysis:	146
Output:	0
WrongServer:	0
IllBkGrnd:	0
NoServer:	0
Timeout:	44
Internal:	0
Unknown:	0

Current

Pointer:	315
Errors:	6
Fraction:	1.20 %

Figure 3.3: RD Statistics panel

The `Current` group of parameters is related to the error rate of the last 500 analyses. The fraction number is used to determine the state and status for the FSM. The number of `Errors` should not be confused with the `cfault` parameter of Figure 3.2. The first is the number of errors of the last 500 analyses, the latter the number of errors of the current sequence. The `Total` group shows the daily statistic parameters since the last Saclay sequence, i.e. the total number of analyzed channels, the number of errors and separated by type, since around midnight. At the end of the Saclay sequence, these parameters are reset to zero.
NB: Keep in mind that there are two sets of statistic variables: one per sequence and one per day!

3.2 RDStore2ConfDb

The RDStore2ConfDb control manager, running on all production PCs with logical number #21, is a more or less obsolete manager. Despite its name, it does not store the channel parameters to a 'real' database, but to a file. It was developed during the installation of the alignment system, when no real configuration database (ConfDb) existed and designers wanted to keep track of the changes applied to the parameters. The manager is in hibernation for almost the entire day, but each night around 02:00 a.m. it becomes active and stores for each channel its parameters (i.e. the camera and light-source addresses, the I^2C parameters and analysis parameters) into a file: one line per channel. These files can be found in the data folder of the PVSS project and have the following format: `“Balx_yymmdd_hhmmss.cfg”` with *x* the server, `“yymmdd”` as date and `“hhmmss”` as time.
Since the introduction of a 'real' ConfDb, the purpose of this manager is redundant. However, its overhead is minimal and its use could be useful whenever the ConfDb is not available. In case a PVSS project has to be built up from scratch, including the creation of the channel datapoints and setting the parameters, there are two ways to do so. First by means of the ConfDb and second by means of the latest saved configuration file generated by this manager.

3.3 RDsvWatchdog

The RDsvWatchdog control manager, running on the super-visor only (i.e. PC9) with logical number #81, determines the states of the barrel alignment FSM. In order to do so, it has to be able to read the datapoints of the remote PVSS projects of the production PCs. For that purpose the following lines are added to its config file:

```
[dist]
distPeer = "pcatlmdtbal1.cern.ch" 91
distPeer = "pcatlmdtbal2.cern.ch" 92
distPeer = "pcatlmdtbal3.cern.ch" 93
distPeer = "pcatlmdtbal4.cern.ch" 94
distPeer = "pcatlmdtbal5.cern.ch" 95
distPeer = "pcatlmdtbal6.cern.ch" 96
distPeer = "pcatlmdtbal7.cern.ch" 97
distPeer = "pcatlmdtbal8.cern.ch" 98
```

The relevant datapoints are read in a loop (polling), instead of using a so-called *dpConnect*. The latter is an asynchronous way of reading a datapoint by means of a call-back routine, which may have its drawbacks by the overhead and complexity involved. Reading the datapoint in an infinite loop guarantees a better determination, especially when a remote PVSS project does not respond anymore (a dpConnect does not detect it). For each PVSS project or service [*x*] the following datapoints are read during each cycle:

Datapoint	Variable	Description
pcatlmdtbal*x*:rdSeq*x*.state	bRunState	run-state of RDSequence
pcatlmdtbal*x*:rdSeq*x*.nPos	dNpos	pointer inside current sequence
pcatlmdtbal*x*:rdSeq*x*.statistics.errFraction	fFraction	error fraction of last 500 analyses
pcatlmdtbal*x*:rdDbase*x*.state	bDbState	state of database storage
pcatlmdtbal*x*:rdDbase*x*.storage	bDbStore	storage active

Table 3.1: State & status datapoints

Besides the datapoints of table X, a set of general datapoints is read, which are meant for the determination of the status. They are known as the watchdog variables of the super-visor. Table Y shows them, including their default value. At startup these variables are read from the configuration file (see x.x) and if not set, given the default value.

Datapoint	Variable	Default	Description
pcatlmdtbal9:rdWatchdog.looptime	dLooptime	30	cycle time
pcatlmdtbal9:rdWatchdog.warnTo	dWarnTo	10	warning timeout
pcatlmdtbal9:rdWatchdog.errTo	dErrTo	20	error timeout
pcatlmdtbal9:rdWatchdog.warnRatio	fWarnRatio	2.0	warning fraction
pcatlmdtbal9.rdWatchdog.errRatio	fErrRatio	5.0	error fraction

Table 3.2: Watchdog variables

4 Appendices

4.1 ConfDb Access

The following SQL queries are possible:

- Get the channel list of server X [1-8]:
```
SELECT * FROM TABLE (atlas_conf_mdt.align_r.get_channel_for_camsrv(X));
```

- Get the values of the parameters of a particular *channel*:
```
SELECT * FROM TABLE (atlas_conf_mdt.align_r.get_values_of_channel('channel'));
```

- Remove (invalidate) a particular *channel*:
```
CALL atlas_conf_mdt.align_w.invalidate_channel('channel');
```

- Create a new *channel*:
```
CALL atlas_conf_mdt.align_w.create_new_channel('channel');
```

- Set a *parameter* of a particular *channel*:
```
CALL atlas_conf_mdt.align_w.set_param_of_channel('channel','parameter','value');
```

Valid parameters are:

analtype	fin14	fin7	nin16	nin9
antiblooming	fin15	fin8	nin17	normalbacklit
autogain	fin16	fin9	nin18	pixelmode
background	fin17	fine	nin19	pixeloffset
binairisation	fin18	forceblack	nin2	pixthreshold
blackreference	fin19	gain	nin20	rAnalog
cammas	fin2	horshuffle	nin21	rCoarse
camras	fin20	inhibitblack	nin22	rFine
camsrv	fin21	ledmas	nin23	rGain
camtop	fin22	ledras	nin24	rLower
chanid	fin23	ledsrv	nin25	rSetup1
coarse	fin24	ledtop	nin26	rSetup2
counter	fin25	linearmode	nin27	rSetup3
div0	fin26	lower	nin28	rUpper
div1	fin27	nimgs	nin29	readmode
enableauto	fin28	nin0	nin3	shufflemode
enablesno	fin29	nin1	nin30	upper
fin0	fin3	nin10	nin31	vershuffle
fin1	fin30	nin11	nin4	whitethreshold
fin10	fin31	nin12	nin5	
fin11	fin4	nin13	nin6	
fin12	fin5	nin14	nin7	
fin13	fin6	nin15	nin8	

4.2 Auto logon and start of Rasdim

At startup of a production PC the user ‘balign’ has to be automatically logged on and the Rasdim server started. Inside the registry the following parameters have to be set:

Variable	Value
DefaultUserName	balign
DefaultPassword	*********
DefaultDomainName	PCATLMDTBAL*x*
AutoAdminLogon	1
ForceAutoLogon	0

Table 4.1: Auto logon parameters

The password is for obvious reasons not shown. The *regedit* application/browser may be used to check/set these settings. The location of these variables inside the registry is:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
```

A Visual Basic script, shown in Script 4.1, sets the necessary parameters automatically.

```
'------------------------------------------------------------
'Autologon
'------------------------------------------------------------

Dim strUser, strPW

strUser = "balign"
strPW = "*********"

Dim objShell, objFSO
Set objShell = WScript.CreateObject("WScript.Shell")
Set objFSO = CreateObject("Scripting.FileSystemObject")

Const HKEY_LOCAL_MACHINE = &H80000002
Dim strComputer, objRegistry, strKeyName

strComputer = "."
Set objRegistry = GetObject("winmgmts:{impersonationLevel=impersonate}!\\" & strComputer &
"\root\default:StdRegProv")
Set fs = WScript.CreateObject("WScript.Network")

computerName = fs.ComputerName

Dim intRet

strCADPath = "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"

intRet = objRegistry.SetStringValue(HKEY_LOCAL_MACHINE, strCADPath, "DefaultUserName", strUser)
intRet = objRegistry.SetStringValue(HKEY_LOCAL_MACHINE, strCADPath, "DefaultPassword", strPW)
intRet = objRegistry.SetStringValue(HKEY_LOCAL_MACHINE, strCADPath, "DefaultDomainName",
computerName)
intRet = objRegistry.SetStringValue(HKEY_LOCAL_MACHINE, strCADPath, "AutoAdminLogon", "1")
intRet = objRegistry.SetStringValue(HKEY_LOCAL_MACHINE, strCADPath, "ForceAutoLogon", "0")


WScript.Quit(0)
```

Script 4.1: setBALAutologon.vbs

In order to start the Rasdim server automatically, the executable has to be added to the startup menu of the ‘balign’ user. The Visual Basic script, shown in Script 4.2, takes care of this. Beware of the version number of Rasdim.

```
Set oFSO = WScript.CreateObject("Scripting.FileSystemObject")

 if oFSO.FolderExists("C:\Documents and Settings\balign\Start Menu\Programs\Startup") = true then
   if oFSO.FileExists("C:\Documents and Settings\balign\Start Menu\Programs\Startup\Rasdim64.lnk") = false then
     if oFSO.FileExists("C:\Balign\bin\Rasdim64\Rasdim64.exe") = true then

       set objShell = Wscript.CreateObject("WScript.Shell")
       set objShortcut = objShell.CreateShortcut("C:\Documents and Settings\balign\StartMenu\Programs\Startup\Rasdim64.lnk")
       with objShortcut
                 .TargetPath = "C:\Balign\bin\Rasdim64\Rasdim64.exe"
                 .Description = "Rasdim64 Server"
                 .WorkingDirectory = "C:\Balign\bin\Rasdim64"
                 .IconLocation = "C:\Balign\bin\Rasdim64\Rasdim64.exe, 0"
                 .Save
       end with
     End If
   End If
 End if
```

Script 4.2: setBALAutostartRasdim.vbs

Both scripts should be executed at least once, in order to ensure automatic logon of ‘balign’ and start of the Rasdim server at startup.

4.3 Rasdim initialization files

The Rasdim server requires two initialization files. The location of them is the same as where the Rasdim executable resides (`C:\Balign\bin\Rasdim64`). One is used to setup the frame-grabber and is called: `synchr2.ccf` and shown in File 4.1. The content of this file is more or less sacred! Do **not** change any value. Especially the variables indicating the width (= 392) and height (=292) are very important. Also the variable `PixelClockSource` is for the alignment system vital. If not set to 2 (indicating an external clock source is applied) the Rasnik system will not work. But the other values should not be altered as well. A lot of them are interrelated.

```
// Data Translation, Inc - Control Config File v1.0

AcquireType         = 0
ActiveLUT           = 0
ActiveLineCount     = 292
ActivePixelCount    = 392
AuthorName          = "HLG"
BoardType           = "DT3162"
Brightness          = 147
ClampEnd            = 48
ClampStart          = 44
Contrast            = 52
Description         = "synchronous read out 2"
ExposeEnabled       = 0
ExposePolarity      = 1
ExposeStartLine     = 1
ExposeStopLine      = 100
FirstActiveLine     = 20
FirstActivePixel    = 74
HSyncInPolarity     = 1
HSyncOutPulseWidth  = 60
InputLineEventMask  = 0
LineFrequency       = 15625
OverlayBitmapFile   = ""
OverlayColorKey     = 0
OverlayEnabled      = 0
PixelClockSource    = 2
RoiHeight           = 292
RoiLeft             = 0
RoiTop              = 0
RoiWidth            = 392
StrobeEnabled       = 0
StrobePolarity      = 1
StrobeStartLine     = 1
StrobeStopLine      = 100
SyncInSource        = 0
SyncOutEnabled      = 0
SyncOutPolarity     = 1
Timeout             = 3
TotalLinesPerFrame  = 625
TotalPixelsPerLine  = 472
TriggerTransition   = 1
TriggerType         = 0
VSyncDelay          = 50
VSyncInPolarity     = 1
VSyncOutPulseWidth  = 3
VideoInputSource    = 0
```

File 4.1: synchr2.ccf

The behavior of the Rasdim server is dependent of a set of so-called configuration variables. All of them can be altered by means of the configuration file `Rasim.ini`. In File 4.2 the current initialization file of PC6 is shown. The content of the other Rasdim.ini files on the other PCs is equal, except of course the value of the configuration variable `LogicServer`. This variable defines the difference between the Rasdim servers on each PC (see also Table 1.4).

```
[RASDIM]
DnsHost=PCATLMDTBAL9.cern.ch
LogicServer=6
LogDir="C:\Results\Log"
Debug=1
Comport=1
GrabDelay=600
RasnikAnal=".\Analysis240.dll"
SpotAnal=".\CalculSaclay520.dll"
FoamAnal=".\Foam.dll"
ResultDir="C:\Results\Balign"
ResultPrefix="rd."
ImageDir="C:\Results\Balign\Images\Sav"
BadImageDir="C:\Results\Balign\Images\Bad"
```

File 4.2: Rasnik.ini of PC6

A default value is assigned whenever a configuration variable is not defined in Rasdim.ini file. Table 4.2 shows the complete list of configuration variables, together with there default value.

Variable	Default value
`DnsHost`	`DIM_DNS_NODE` or `“localhost”`
`LogicServer`	`99`
`LogicDevice`	`0`
`LogDir`	`“.”`
`Debug`	`1`
`Comport`	`1`
`Baudrate`	`115200`
`GrabDelay`	`1000`
`RasnikAnal`	`“.\Analysis240.dll”`
`SpotAnal`	`“.\CalculSaclay512.dll”`
`FoamAnal`	`“.\Foam.dll”`
`ResultDir`	`“D:\Results”`
`ResultPrefix`	`“Ras”`
`ImageDir`	`“D:\Results\Image\sav”`
`BadImageDir`	`“D:\Results\Image\bad”`

Table 4.2: Rasdim configuration variables

- **DnsHost**: the name of the machine on which the DNS server is running. If not set in the Rasnik.ini file then the environment variable DIM_DNS_NODE is taken. If this one is not defined, the value “localhost” is taken.
- **LogicServer**: for the barrel alignment readout system, only values $1 \le x \le 8$ are possible.
- **LogicDevice**: only used when multiple frame-grabbers in a single PC are installed.
- **LogDir**: the folder in which Rasdim puts its logging files.
- **Comport**: the serial port number to which the TopMux is connected.
- **Baudrate**: the communication speed on the serial link to the TopMux (default = 115.2 kBits/s).
- **RasnikAnal**: the analysis module for type Rasnik (#0)
- **SpotAnal**: the analysis module for type Spot (#1)
- **FoamAnal**: the analysis module for type Foam (#2)
- **ResultDir**: the folder in which Rasdim places the result files
- **ResultPrefix**: the result files are prefixed by this variable
- **ImageDir**: the folder in which the analysed images are stored (only if auxiliary command AUX_SAVE_IMAGE is set).
- **BadImageDir**: the folder in which an image is store if the analysis failed (only if the auxiliary command AUX_DONT_SAVE_BADIMAGE is not set).

4.4 Quick Facts

Channels	Rasnik: Spot: Total:	5296 516 **5812**
Image size (pixels)	Width: Height: Size:	392 292 114464
USA15	Rack:	Y.28-19.A1
PCs	DELL **PE750** rack-mountable, 1U	3.2 GHz, 1 GB RAM, 80 GB harddisk
KVM-switch	DELL **2161DS**	
Frame-grabber	Data-Translation: **DT3162**	Driver version: 1.1.2.3
TopMux	Firmware version:	YMD: 05/08/30

Table 4.3: Quick facts