TITLE : PCON Software Design Document

DOCUMENT NUMBER : 3140AE0022
ISSUE: B

SYNOPSIS : This document describes the software design of the PCON software of the PFIS.

KEYWORDS : PFIS, RSS, PDET

PREPARED BY : Anthony Koeslag

APPROVED : 

DATE : 26 February 2007

This issue is only valid when the above signatures are present.
Printed: 27/02/07 13:44
**ACRONYMS AND ABBREVIATIONS**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATP</td>
<td>Acceptance Test Procedure</td>
</tr>
<tr>
<td>ATR</td>
<td>Acceptance Test Report</td>
</tr>
<tr>
<td>BMS</td>
<td>Building Management System</td>
</tr>
<tr>
<td>CDR</td>
<td>Critical Design Review</td>
</tr>
<tr>
<td>CIN</td>
<td>Code Interface Node (a LabVIEW function to interface to other SW)</td>
</tr>
<tr>
<td>ELS</td>
<td>Event Logger Software</td>
</tr>
<tr>
<td>EDS</td>
<td>Environmental Display System</td>
</tr>
<tr>
<td>HET</td>
<td>Hobby-Eberly Telescope</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output (Device)</td>
</tr>
<tr>
<td>ICD</td>
<td>Interface Control Dossier</td>
</tr>
<tr>
<td>MMI</td>
<td>Man-Machine Interface</td>
</tr>
<tr>
<td>OPT</td>
<td>Operational Planning Tool</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PDR</td>
<td>Preliminary Design Review</td>
</tr>
<tr>
<td>PFIS</td>
<td>Prime Focus Imaging Spectrograph</td>
</tr>
<tr>
<td>PI</td>
<td>Principal Investigator (Astronomer)</td>
</tr>
<tr>
<td>PIPT</td>
<td>PI Planning Tool</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable-Logic Controller</td>
</tr>
<tr>
<td>PMAS</td>
<td>Primary Mirror Alignment System</td>
</tr>
<tr>
<td>SA</td>
<td>SALT Astronomer</td>
</tr>
<tr>
<td>SALT</td>
<td>Southern African Large Telescope</td>
</tr>
<tr>
<td>SAMMI</td>
<td>SA Machine Interface</td>
</tr>
<tr>
<td>SC</td>
<td>Software Component (e.g. part of the TCSS)</td>
</tr>
<tr>
<td>SCAM</td>
<td>Salticam (Acquisition camera)</td>
</tr>
<tr>
<td>SCL</td>
<td>SALT Command Language (sent to TCSS)</td>
</tr>
<tr>
<td>SDB</td>
<td>Science Database</td>
</tr>
<tr>
<td>SDD</td>
<td>Software Design Document</td>
</tr>
<tr>
<td>SDP</td>
<td>Software Development Plan</td>
</tr>
<tr>
<td>SI</td>
<td>Software Item (the TCSS is a Software Item)</td>
</tr>
<tr>
<td>SO</td>
<td>SALT Operator</td>
</tr>
<tr>
<td>SOMMI</td>
<td>SO Machine Interface</td>
</tr>
<tr>
<td>SRS</td>
<td>Software Requirement Specification</td>
</tr>
<tr>
<td>STARCAT</td>
<td>Object Catalogue</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>TBC</td>
<td>To Be Confirmed</td>
</tr>
<tr>
<td>TBD</td>
<td>To Be Determined</td>
</tr>
<tr>
<td>TCS</td>
<td>Telescope Control System</td>
</tr>
<tr>
<td>TCSS</td>
<td>TCS Server</td>
</tr>
<tr>
<td>TPM</td>
<td>Telescope Pointing Machine (software for Astrometric Pointing)</td>
</tr>
<tr>
<td>VI</td>
<td>Virtual Instrument (LabVIEW function)</td>
</tr>
<tr>
<td>WEB</td>
<td>SALT web-server</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

1. Scope ......................................................................................................................... 5
2. Referenced Documents ............................................................................................... 5
3. Feature definition ......................................................................................................... 5
4. Previously Existing software ..................................................................................... 5
5. Concept definitions ...................................................................................................... 5
5.1 Object Data Manager (ODM) .................................................................................. 5
5.2 Queues ...................................................................................................................... 6
5.3 Action and Command Arrays .................................................................................. 6
5.4 Modules .................................................................................................................... 6
6. PCON Main Software overview ............................................................................... 6
7. PCON Communications Network ............................................................................ 7
8. PCON Main top level .................................................................................................. 8
  8.1 State Machine ......................................................................................................... 8
  8.1.1 Sates ................................................................................................................... 9
  8.1.2 Control Arbitration ............................................................................................ 11
8.2 Execution Engine ...................................................................................................... 12
  8.2.1 Action Array Manager ...................................................................................... 13
  8.2.2 Command Generator ....................................................................................... 14
  8.2.3 Command Array Manager ............................................................................... 14
9. PCON Managers ......................................................................................................... 14
  9.1 Status Manager ...................................................................................................... 15
  9.2 TCS Manager ........................................................................................................ 16
  9.3 SCL Manager ........................................................................................................ 16
  9.4 PDET Manager ...................................................................................................... 16
  9.5 MMI Manager ........................................................................................................ 16
  9.6 PXI Manager .......................................................................................................... 16
  9.7 Error Manager ....................................................................................................... 17
  9.8 Log Manager .......................................................................................................... 17
  9.9 File Manager .......................................................................................................... 17
10. PCON PXI Sub-managers ....................................................................................... 17
  10.1 Articulation_Angle sub-manager ......................................................................... 18
  10.2 Beam_Splitter sub-manager ............................................................................... 18
  10.3 Etalons sub-manager ............................................................................................ 18
  10.4 Filter sub-manager ............................................................................................... 18
  10.5 Focus sub-manager .............................................................................................. 18
  10.6 Grating sub-manager ........................................................................................... 18
  10.7 Grating_Angle sub-manager ............................................................................... 18
  10.8 Shutter sub-manager ............................................................................................ 18
  10.9 Slit mask sub-manager ........................................................................................ 18
  10.10 Wave_plate sub-manager ................................................................................... 18
11. Remote MMI ............................................................................................................. 19
  11.1 State Machine ...................................................................................................... 19
  11.1.1 States ............................................................................................................... 19
  11.2 Execution Engine ................................................................................................ 20
  11.3 User Interface Manager ...................................................................................... 20
  11.4 PCON Manager ................................................................................................... 20
  11.5 File Manager ....................................................................................................... 20
12. APENDIX: Configuration Command Sequences ............................................... 20
  12.1 Configuration Logic ............................................................................................. 20
12.2 Configuration Sequences defined ........................................................................... 21
12.2.1 Configuration Sequence 1 .................................................................................. 22
12.2.2 Configuration Sequence 2 .................................................................................. 22
12.2.3 Configuration Sequence 3 .................................................................................. 22
12.2.4 Configuration Sequence 4 .................................................................................. 22
12.2.5 Configuration Sequence 5 .................................................................................. 23

13 APENDIX: Procedure Sequences defined ................................................................. 23
13.1 Exposure procedure ............................................................................................... 23
13.2 Polarimetric Imaging and Spectroscopy ................................................................. 23
13.3 Fabry-Perot Imaging ............................................................................................... 24
13.4 Polarimetry Fabry-Perot Imaging ............................................................................ 24

14 APENDIX: List of Queues and arrays used in PCON Main .................................... 24
14.1 Action Array ........................................................................................................... 24
14.2 Command Array ..................................................................................................... 24
14.3 Q-event .................................................................................................................. 24
14.4 Q-scl ..................................................................................................................... 25
14.5 Q-pxi ..................................................................................................................... 25
14.6 Q-pdet .................................................................................................................. 25
14.7 Q-mmi .................................................................................................................. 25
14.8 Q-error ................................................................................................................. 25
14.9 Q-log .................................................................................................................... 25
14.10 Q-file .................................................................................................................. 25
14.11 Q-command Status .............................................................................................. 25
14.12 Q-action status ..................................................................................................... 25
14.13 Q-<PXI sub-manager> ......................................................................................... 25

15 APENDIX: Initialisation File ...................................................................................... 25

16 APENDIX: Previous PCON software design document: ....................................... 28
16.1.1 Southern African Large Telescope ..................................................................... 28
16.1.2 Prime Focus Imaging Spectrograph .................................................................... 28
16.1.3 Control System Software Design Document ...................................................... 28
16.1.4 SALT-3140AE0022 ........................................................................................... 28
1 Scope

This document describes the design of the PCON software.

2 Referenced Documents

The following documents are related to this design document and are applicable to the extent mentioned herein.

1000AB0044 SALT LabVIEW Coding Standard
1000AS0040 SALT Operational Requirements
SALT-3170AE0002 Prime Focus Imaging Spectrograph Operations Concepts Definition Document
1000AE0033 PFIS ICD design report
SALT-3140AE0001 Prime Focus Imaging Spectrograph Preliminary Control System Design
SALT-3140AE0019 Prime Focus Imaging Spectrograph Control System Design Philosophy
SALT-3140AE0022 Prime Focus Imaging Spectrograph Control System Software Design Document
SALT-3140AE0026 Prime Focus Imaging Spectrograph Roadmap to the PFIS Control System Software
SALT-3140AS0015 Prime Focus Imaging Spectrograph Interlock Specification and Design Document
3140AS0027-0001 PCON Software Requirements Document

3 Feature definition

(See 3140AS0027-0001 PCON Software Requirements)

1. PCON provides a user interface to PFIS.
2. PCON provides a TCS interface to PFIS.
3. PCON controls the PFIS hardware, for setting up hardware configurations.
4. PCON sends PFIS detector settings to PDET.
5. PCON coordinates science procedures.
6. PCON has a set of engineering controls and indicators as needed for technical purposes.
7. The PCON software forms a communications and control layer between the TCS and PDET.

4 Previously Existing software

1. TCS
2. PDET
3. Fits Header generation (expanded to include new data)
4. Procedure control (expanded to include new procedures)
5. Hardware control (extended to incorporate parallel command execution)
6. PXI control software (minor modifications)
7. High level control of hardware systems: Slit mask, Focus, Grating, articulation, filters, wave plates, shutter, etalons, and beam splitter.

5 Concept definitions

5.1 Object Data Manager (ODM)

Object Data Managers (ODMs) are used throughout the PCON software for distributing data between modules. Each module writes to its own ODMs, any other module that needs that information can read from them at any stage.

ODMs consist of a loop (run once per call) and register (part of the loop). Data can be written to the register using a 'put' command. Data is read using a 'get' command.

ODMs can be thread safe for allowing data to be written from multiple modules without loosing
data if two modules try to write at the same time.

5.2 Queues

Queues are used when one Module is to receive data from one or more modules where the order it is added to the queue is important. Queues work as First in First off. Once an item on the queue is read it is stripped off the queue. It is possible to read all the items on the queue without stripping them off.

5.3 Action and Command Arrays

Some actions can be executed in parallel, some actions require specific actions to occur before they can be executed. Actions sent from the State Machine to the Execution Engine are called Actions in this document. Actions generated in the Execution Engine based on these Actions are called Commands in this document. Apart from this distinction Action and Command arrays are identical in use and function. Commands and actions have an associated state which can be:

1. Waiting – The command is waiting for other commands to complete.
2. Ready – There are no more constraints to be met and the command will be sent to the relevant manager.
3. Sent – The command has been sent to the relevant manager.
4. Executing – The command is in the process of executing.
5. Complete – The command complete without error.
6. Aborted – The command was aborted before it could complete.
7. Failed – An error occurred.

Each Command has a unique ID number. Each command has a list of command IDs that must be completed before its state changes from Waiting to Ready. If a command has no dependencies then this array will be empty.

5.4 Modules

A software module performs a set of separable tasks. The PXI manager is a module that receives all PXI related commands, processes them and returns feedback on the commands as well as data received from the hardware. The File manager module controls all file access. The Execution Engine coordinates all high level command execution. And so on.

6 PCON Main Software overview

PCON Main is made up of a State Machine, Execution Engine and various Managers. Each manager handles a separable set of functions. Each Manager can have Sub-Managers which handle sub-sets of functions inside a manager.
7 **PCON Communications Network**

This diagram shows how PCON communicates with various systems on SALT.

![PCON Network Communications Diagram](image)

**Figure 2 PCON Network Communications diagram**
8 PCON Main top level

The PCON Main program is the heart of the PCON software. Separated from the user interface, the Remote MMI, PCON Main handles commands and events from TCS and the Remote MMI. Determines the actions needed to respond to the commands and events, and coordinates the execution of these actions.

At the top level PCON Main is made up of a State Machine and the Execution Engine. Lower more specific functions are handled by various managers and sub-managers.

![State Machine Diagram]

8.1 State Machine

Items on the event queue, added from any of the modules, are processed by the State Machine. The State Machine decides what events can be processed, based on the current PCON State and status values. Then appropriate are added to the Actions Array. The actions on the Action Array can be simple, like apply configuration, or more complex like abort procedure then apply configuration then run procedure. Actions can be dependent on other actions completing or independent. Dependent actions will have a list of all the action ids that must complete before they can execute.
8.1.1 States

The State Machine will use the states to determine what commands are allowed and to report to TCS what PCON is currently busy doing.
Table 1 State Change Events

<table>
<thead>
<tr>
<th>Transition</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PCON is commanded to Initialisation</td>
</tr>
<tr>
<td>2</td>
<td>Software and hardware initialisation completes and has received a command to Ready</td>
</tr>
<tr>
<td>3</td>
<td>PCON is commanded to Shutdown</td>
</tr>
<tr>
<td>4</td>
<td>Software and hardware shutdown completes and mode changes to Off</td>
</tr>
<tr>
<td>5</td>
<td>Positioning command event</td>
</tr>
<tr>
<td>6</td>
<td>Positioning command is completed and has received a command to Ready</td>
</tr>
<tr>
<td>7</td>
<td>Procedure command event</td>
</tr>
<tr>
<td>8</td>
<td>Procedure command is completed and has received a command to Ready</td>
</tr>
<tr>
<td>9</td>
<td>PCON is commanded to Engineering mode</td>
</tr>
<tr>
<td>10</td>
<td>PCON is commanded to Initialisation</td>
</tr>
<tr>
<td>11</td>
<td>PCON is commanded to Engineering</td>
</tr>
<tr>
<td>12</td>
<td>PCON is commanded out of Engineering and reverts back to Major Fault</td>
</tr>
<tr>
<td>13</td>
<td>PCON is commanded to Kill</td>
</tr>
<tr>
<td>14</td>
<td>PCON is commanded to Kill</td>
</tr>
<tr>
<td>15</td>
<td>Kill completes and mode changes to Off</td>
</tr>
<tr>
<td>E</td>
<td>A fault occurs requiring attention before further PFIS operation can continue</td>
</tr>
<tr>
<td>R</td>
<td>The current fault has been fixed and PCON returns to the previous mode</td>
</tr>
</tbody>
</table>

Table 2 State actions

<table>
<thead>
<tr>
<th>State</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Initialisation files are loaded. Control arbitration takes place. PCON can be commanded to Initialisation state.</td>
</tr>
<tr>
<td>Initialisation</td>
<td>The hardware is initialised: power is turned on hardware state detection algorithm is run. Constant files are re-loaded PCON can be commanded to Ready state. This mode change will take effect when the initialisation is complete.</td>
</tr>
<tr>
<td>Ready</td>
<td>PCON waits for commands PCON can be commanded to Configure; Procedure; Engineering; Shutdown and Kill States.</td>
</tr>
<tr>
<td>Shutdown</td>
<td>The hardware is parked. The hardware is powered down. PCON can be commanded to Off state. This mode change will take effect when the shutdown is complete.</td>
</tr>
<tr>
<td>Configure</td>
<td>PCON is applying the configuration settings. PCON can be commanded to Ready state. This mode change will take effect when the configuration is complete.</td>
</tr>
<tr>
<td>Procedure</td>
<td>A Procedure is being executed. PCON can be commanded to Ready state. This mode change will take effect when the procedure is complete.</td>
</tr>
<tr>
<td>Engineering</td>
<td>Major Fault is ignored. Science controls are locked out on the MMI. Engineering controls are unlocked on the MMI. Very low level commands can be received in this state. To return to ready mode Initialisation can be commanded or the user can release Engineering mode which will return the software to Major Fault mode. PCON can be initialisation state or released back to Major Fault.</td>
</tr>
</tbody>
</table>
Major Fault | An event has occurred that could be dangerous to the hardware or could prevent further science. PCON can be commanded to Initialisation, Kill, and Engineering states.
---|---
Kill | The software and hardware is stopped without being parked. The hardware is powered down. PCON will automatically change to Off state when complete.

8.1.2 Control Arbitration

After PCON starts it reports its external state as Maintenance and its internal state as Off. If TCS is running it requests and is granted control of PCON. PCON reports its external state the same as its internal state.

If PCON does not request control, a Remote MMI may request control. PCON will grant the MMI control and continue to report its external state as Maintenance.

If TCS now requests control the request will be ignored.

Control can be released from the Remote MMI, at this point TCS or another remote MMI can request control.

![Diagram of Control Arbitration](image)

**Figure 6 Control Arbitration**

<table>
<thead>
<tr>
<th>Transition</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PCON has started. The external state = Maintenance The Internal state = Off</td>
</tr>
<tr>
<td>2</td>
<td>TCS requests and is granted control.</td>
</tr>
</tbody>
</table>
The external state is reported as what the internal state is reporting. The internal state changes as per state change commands.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>TCS releases control</td>
</tr>
<tr>
<td></td>
<td>The external state = Maintenance</td>
</tr>
<tr>
<td>4</td>
<td>A remote MMI requests control</td>
</tr>
<tr>
<td></td>
<td>The external state = Maintenance</td>
</tr>
<tr>
<td></td>
<td>The internal state changes as per state change commands</td>
</tr>
<tr>
<td>5</td>
<td>A remote MMI releases control</td>
</tr>
<tr>
<td></td>
<td>The external state = Maintenance</td>
</tr>
<tr>
<td>6</td>
<td>A different remote MMI requests control</td>
</tr>
<tr>
<td></td>
<td>The external state = Maintenance</td>
</tr>
<tr>
<td></td>
<td>The internal state changes as per state change commands</td>
</tr>
<tr>
<td>7</td>
<td>A different remote MMI releases control</td>
</tr>
<tr>
<td></td>
<td>The external state = Maintenance</td>
</tr>
</tbody>
</table>

### 8.2 Execution Engine

The Execution Engine will turn actions into sequences of commands and manage the execution order of the commands. The Execution Engine will allow some commands to execute in parallel to some commands. Some commands require specific conditions to be met before they can complete.
8.2.1 Action Array Manager

A list of Actions is received from the State Machine. An action can require other actions to complete before they execute or can be immediately available for execution.

An example of an action that can be immediately available to execution would be an Abort action. This is immediately converted to an Abort command and processed.

An example of dependant actions that are depended would be if TCS commands a configuration followed by a procedure.
The configure action depends on the PCON state being commanded to Configure state and is complete when all the resulting commands are complete.
The procedure action depends on the configure action completing and PCON being commanded to Procedure state and is complete when all the resulting commands complete.

Step by step this works as follows:
1. TCS commands PCON to Configure State
2. A sequence of commands is generated to satisfy the configuration action.
3. The Configuration action is complete when all the related commands are completed.
4. PCON changes to Ready State when TCS commands it to Ready.
5. When TCS sees the Ready State it commands PCON to Procedure State.
6. A sequence of commands is generated to satisfy the procedure action.
7. The Procedure action is complete when all the related commands are complete.
8. PCON changes to Ready state when TCS has commanded it to Ready.

8.2.2 Command Generator

The Command Generator determines the commands needed to complete an actions. An action like the configure action will generate a different sequence of commands based on the current configuration of the hardware. Simplistically the configure action could be a no operation if the hardware is already configured. Complexly the configure action could generate a sequence of commands that first remove etalons before rotating the grating and articulation while parallel commands are being processed to change the slitmasks. Some actions will call for current commands to be aborted. The command Generator will make these decisions and add the correct commands to the command array.

8.2.3 Command Array Manager

The command Array Manager checks the state of each command and determines when commands can change from `waiting` to `ready`. When commands are `complete`, `aborted` or `failed` they are removed from the command array. A history of all commands and their status are logged.

When a command state changes to `ready` the command is sent to the appropriate manager and its status is changed to `sent`. The manager receives that command and when it allows the command to be processed it reports the state change to `executing` on the command status queue. Similarly when the manager determines that the command has succeeded, failed or been aborted it reports the state change on the command status queue.

9 PCON Managers

The Managers are any software module that managers one of PCONs resources. The SCL manager looks after all commands sent to the TCS as a SCL command, and checks the status of the command. The PXI manager looks after all commands that the PXI software must handle and checks the status of the hardware as read by the PXI software. And so on.
9.1 Status Manager

The Status Manager analyses status from the various modules. Events are generated based on specific conditions. Temperatures exceeding thresholds generate events. Commands from TCS generate events. And so on.

Figure 8 Managers
9.2 TCS Manager

The TCS Manager opens a DataSocket connection to TCS. If this fails an error is reported on
the error queue and after a pause tries to re-establish a connection.
With a successful connection data is constantly written in ICD clusters to DataSocket for TCS to
read.
Data is constantly read from DataSocket which is added to ODMs for other modules to access.
Commands are sent to PCON from TCS on DataSocket. When an item in one of these command
clusters changes an event is added to the event queue.
Data from PCON to TCS is constantly written to DataSocket.
The clusters used for this communication between TCS and PCON are defined in the Interface
Control Dossier, ICD. Any reference to ICD clusters are referring to clusters used for
communication with TCS and its sub-systems.

9.3 SCL Manager

The SCL Manager opens and manages a TCP connection to TCS.
Commands sent to TCS from PCON are sent as SALT Command Language (SCL) commands.
These are text based commands, stored in text files. The command is sent over a TCP connection
to TCS. TCS returns the commands status via the same connection.
The SCL manager reports the command status on the command status queue.

9.4 PDET Manager

The PDET Manager opens and manages a DataSocket connection to PDET.
Commands are sent to PDET via a DataSocket connection. The communication clusters do not form
part of the ICD. Command status is updated on the command status queue.
Status from PDET is read from the DataSocket connection and added to ODMs.

9.5 MMI Manager

The MMI connection is managed by the Remote MMI.
The MMI manager allows Remote MMIs to read data off a local DataSocket. When a Remote MMI is
granted control of PCON the Remote MMI will be allowed to open a TCP connection to the PCON
computer.
Commands and data are sent to the Remote MMI via DataSocket.
MMI events are received over a TCP connection.

9.6 PXI Manager

The PXI Manager looks after all commands from PCON to the PFIS hardware. The hardware status
is read and stored in ODMs.
The PXI control software has already been developed and is used as is except for some small changes to allow for dual command execution of the various hardware systems.

The PXI manager receives commands on the PXI queue. The are distributed to the correct sub manager queue. The sub-managers control individual hardware systems and report the status of commands on the command status queue.

9.7 Error Manager

All errors are added to the error queue. The error manager determines if the error indicates a major fault and generates an event to the State Machine. When errors are no longer valid an event is generated and sent to the State machine. Some errors are to be logged, these are sent to the log queue.

9.8 Log Manager

All items to be log-able events are added to the log queue. The log manager determines, according to a file which events to log and sends the file queue.

9.9 File Manager

All file access is handled by the file manager.

10 PCON PXI Sub-managers

The PXI sub-managers individually control a hardware subsystem. The PXI manager sends commands to each sub-manager which then reports the command status on the command status queue. Each sub-manager has a state table which creates a required transition. Numerous transitions are often required to get to the desired state. These sub-managers were already developed and are used as is except for some control changes as need to allow these to run in parallel with other sub-managers.
10.1 **Articulation Angle** sub-manager
Controls the articulation angle on the PFIS instrument.

10.2 **Beam Splitter** sub-manager
Controls the insertion or removal of the beam splitter on the PFIS instrument.

10.3 **Etalons** sub-manager
Controls the insertion or removal of the two etalons on the PFIS instrument.

10.4 **Filter** sub-manager
Controls the insertion or removal of various filters on the PFIS instrument.

10.5 **Focus** sub-manager
Controls the focus position on the PFIS instrument.

10.6 **Grating** sub-manager
Controls the insertion or removal various gratings on the PFIS instrument.

10.7 **Grating Angle** sub-manager
Controls the grating angle on the PFIS instrument.

10.8 **Shutter** sub-manager
Controls the opening and closing of the shutter on the PFIS instrument.

10.9 **Slit mask** sub-manager
Controls the insertion or removal of various slit masks on the PFIS instrument.

10.10 **Wave plate** sub-manager
11 Remote MMI

The Remote MMI is a stand alone piece of software that can be installed and control PCON from any computer within the SALT fire wall. The Remote MMI reads status, and commands, from DataSocket on the PCON computer. The Remote MMI sends commands to PCON via a TCP connection.

11.1 State Machine

The State Machine receives events on the event queue and determines, based on the current state, weather the event can be processed and what actions to take.

11.1.1 States
1. Observer

The MMI updates all indicators but all the controls are locked out. PCON does not consider the Remote MMI in control and commands are not received from this Remote MMI.

2. Science MMI

The user can use the Remote MMI to do science configurations and Procedures.

3. Engineering MMI

The user can use the Remote to do engineering work on PCON.

11.2 Execution Engine

The Execution Engine takes all actions developed by the State Machine and creates commands which are sent to the Managers.

11.3 User Interface Manager

This Manager consists of the front Panel and the event structure which takes all front panel events and puts them on the event queue. This manager also updates status displays and locks out controls that are not allowed by the user at that time.

11.4 PCON Manager

This manager opens and manages a DataSocket and TCP connection, when the Remote MMI is in control, to PCON. All commands to PCON are handled by this manager. All status and commands from PCON are handled.

11.5 File Manager

Reading and writing files is handled by the file manager.

12 APENDIX: Configuration Command Sequences

The Configuration Command Sequences send commands to the PXI sub-managers in a specific order to get the required hardware configuration. The logic for which sequence to use is based on weather an Etalon is in the optical beam or not. The new configuration cluster is compared to the current configuration cluster. In the tables below a T indicates an item in the configuration cluster that has changed. The number in the sequence column indicates which sequence to run, the sequences are listed below.

12.1 Configuration Logic
Etalon was in

<table>
<thead>
<tr>
<th>Articulation</th>
<th>Grating Angle</th>
<th>Grating</th>
<th>Etalon</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td></td>
<td>T</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>Error</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>Error</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>Error</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>Error</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>Error</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>Error</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>Error</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>4</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>5</td>
</tr>
</tbody>
</table>

**Table 4 Configuration Logic, when an Etalon was in.**

Table 4 shows the logic for which sequence to choose when an Etalon was in when the new configuration cluster is received.

Etalon was out

<table>
<thead>
<tr>
<th>Articulation</th>
<th>Grating Angle</th>
<th>Grating</th>
<th>Etalon</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 5 Configuration Logic, when both Etalons were out.**

Table 5 shows the logic for which sequence to choose when an Etalon was in when the new configuration cluster is received.

**12.2 Configuration Sequences defined**
12.2.1 Configuration Sequence 0
This is a non operation. Just defined to support the tables above.

12.2.2 Configuration Sequence 1
The Articulation and Grating angles are changed. The Beam splitter is inserted or removed. There are no dependencies.

12.2.3 Configuration Sequence 2
The Articulation and Grating Angles are rotated to 0. The Beam splitter is inserted or removed. The Grating is changed, dependant on the articulation and grating angles being zero. The Articulation and grating angles are rotated to the desired angles, dependant on the grating change completing.

12.2.4 Configuration Sequence 3
The Articulation and grating angles are rotated to zero, and the beam splitter is inserted or removed. The grating is removed, dependant on the articulation and grating angles being zero. One or both Etalons are inserted as required. Dependant on the grating removal completing.

12.2.5 Configuration Sequence 4
The Etalons and beam splitter are inserted or removed as required, there are no dependencies.
12.2.6 Configuration Sequence 5

The etalons are removed and the beam splitter is inserted or removed. The grating is inserted as required, dependant on the Etalons being removed. The articulation and grating angles are set, dependant on the grating being inserted.

13 APENDIX: Procedure Sequences defined

This section describes the procedures and the generation of the commands for the command array. The wire diagrams represent the dependencies, wiring from left to right, of each command in the command sequence.

13.1 Exposure procedure
This is the basic exposure procedure, the shutter is open for a specified time period and closed again, the image or spectrum is the read out of the CCD. This is used for Imaging and Spectroscopy.

13.2 Polarimetric Imaging and Spectroscopy
An array of HWP and QWP (half wave and quarter wave plate) positions is specified and the procedure runs an exposure procedure for each combination.
13.3 Fabry-Perot Imaging
An array of Etalon positions is specified and the procedure runs an Imaging exposure procedure with each setting.

13.4 Polarimetry Fabry-Perot Imaging
An array of Etalon positions is specified and the procedure does a Polarimetry Imaging and Spectroscopy procedure with each value.

14 APENDIX: List of Queues and arrays used in PCON Main
14.1 Action Array
All actions generated by the State Machine and processed by the Execution Engine are placed on the Action Array. The status of the actions on the Action array are updated via the action status queue from the Command Array Manager in the Execution Engine.

14.2 Command Array
All commands generated by the Command Generator in the Execution Engine are added to the Command Array. The status of each command is updated from the command status queue in the Command Array Manager in the Execution Engine. Command status can be updated on the command status queue from any of the managers that handle the command.

14.3 Q-event
All events generated for the State Machine to process. These events can be generated throughout the PCON Main Modules.
14.4 **Q-scl**
All commands to be sent to TCS are sent as (SALT Command Language) SCL commands. All commands to be set via the SCL manager are put onto this queue. The Execution Engine is the only module to add items to this queue.

14.5 **Q-pxi**
All commands to be processed by the PXI manager are placed on this queue. The Execution Engine is the only module to add items to this queue.

14.6 **Q-pdet**
All commands to be processed by the PDET manager are placed on this queue. The Execution Engine is the only module to add items to this queue.

14.7 **Q-mmi**
All commands to be processed by the Remote MMI manager are placed on this queue. The Execution Engine is the only module to add items to this queue.

14.8 **Q-error**
All error events to be handled by the Error manager are placed on this queue. Any module can add items to this queue.

14.9 **Q-log**
All error events to be handled by the Log manager are placed on this queue. Any module can add items to this queue.

14.10 **Q-file**
All commands to be handled by the File manager are placed on this queue. The Error Manager, Log Manager and Event Engine will add items to this queue.

14.11 **Q-command Status**
All command statuses will be reported to the Execution Engine on this queue. All modules will add items to this queue.

14.12 **Q-action status**
All action statuses will be reported to the Action manager in the Execution Engine on this queue. The Command Manager in the Execution Engine will add items to this queue.

14.13 **Q-<PXI sub-manager>**
All the PXI sub-managers have queues for receiving commands from the PXI manager. The PXI manager processes items on the Q-pxi queue and places them on the appropriate queue for the PXI sub-manager.

15 **APENDIX: Initialisation File**

A configuration file will be used to store various configurations
Settings.

The following are settings that have already been determined, though could change in the future:

Slitmask Constants
- Steps per Turn=200
- Turns per Inch=8
- Station Break=11
- Wide Station (mm)=9.000000
- Narrow Station (mm)=5.800000
- Station 1 (steps)=2731
- Elevator Home (volts)=8.590000
- Encoder Constant (volts/mm)=0.02504
- Home Trim (steps)=570
- Station Trim (steps)=-5

Waveplate Constants
- Bearing Teeth=144
- Pinion Teeth=30
- Steps per Turn=500
- Encoder Counts per Rev=81000
- Stations per Rev=32
- HWP Station 0 (steps)=758
- QWP Station 0 (steps)=1321

Focus Constants
- Steps per Turn=48
- Gear Ratio=28.444444
- Screw Pitch (mm/turn)=0.400000
- LVDT @ Home (volts)=8.672000
- Encoder Constant (V/mm)=-4.000000
- LVDT @ Fwd Limit (volts)=2.6000
- Steps @ Fwd Limit=46676
- LVDT @ Rev Limit (volts)=8.890000

Grating Constants
- Magazine Steps per Turn=500
- Rotate Steps per Turn=800
- Magazine Pitch (mm)=4.000000
- Rotate Degrees per Turn=2.000000
- Rotate Encoder Counts per Turn=2048
- Encoder Constant (Volts/mm)=0.026247
- Magazine Station 1 (steps)=-630
- Magazine Station Width (mm)=30.000000
- Magazine Home (volts)=4.960000
- Rotate Home Trim (steps)=-269
- Magazine Station Trim (steps)=20

Filter Constants
- Steps per Turn=200
- Turns per Inch=20
- Station Width (mm)=14.000000
- Encoder Constant (V/mm)=0.026247
- Station 1 (steps)=-818
- Magazine Home (volts)=1.710000
- Station Trim (steps)=20

Articulation Constants
- Steps per Turn=5000
- Gear Box Ratio=40
- Pinion Teeth=15
- Rack Teeth=200
- Station Spacing (deg)=0.750000
- Station 1 Offset (deg)=1.750000
- Home Trim (steps)=9
- Encoder Counts per Turn=7684
16 APENDIX: Previous PCON software design document:

16.1.1 Southern African Large Telescope
16.1.2 Prime Focus Imaging Spectrograph
16.1.3 Control System Software Design Document
16.1.4 SALT-3140AE0022

Jeffrey W Percival

Modification Record

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>02-Mar-2003</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>05-Mar-2003</td>
<td>Update reconfig times, add discussion of parallelism</td>
</tr>
</tbody>
</table>

Table of Contents

- Introduction
- Overview
- Environment
- Client/Server Design
- Interlocks
- Server Modes
- State Management
- Observing Procedures
- Error Management
- Data and Command Management
- Communication with Other SALT Subsystems

Introduction

This note describes the software design for the PFIS control system.

Overview

PFIS uses 9 mechanisms in support of its observing states. These mechanisms use a total of 8 stepper motors and 16 pneumatic actuators.

The software must provide the following capabilities:

- Configure mechanisms to support observing states
- Enforce interlock conditions to avoid damage to the instrument & operators
- Provide instrument status data sufficient to support operations, monitor instrument performance, and allow error recovery by the operators

Environment
The PFIS Control System software runs on a standard Windows (currently Windows 2000) PC. The software is written in the National Instruments (NI) graphical programming language LabVIEW. Windows is required for at least part of the control system software because although LabVIEW is supported on Mac OS and Linux, National Instruments currently supports motion control only under Windows.

**Client/Server Design**

We choose a client/server design, in which the user interface is the client, and the server runs the hardware. Each is a stand-alone piece of software, and they communicate over a network using LabVIEW's data socket server (DSS) function. The DSS provides a simple network communications function, and allows both client and server to run on the same CPU or different CPUs without modification of the software.

We also choose the format of the commands passing from the client to the server to be simple text strings (e.g. "mask fetch" or "grating insert"). This allows scripting in a natural way. Human-readable observing scripts will be generated in advance using web-based perl CGI scripts. The PFIS user interface (the client) can then simply read these scripts and forward the commands to the server. The server doesn't need to know about the source of the commands, and no script-dependent software will reside in the server.

Note that choosing a client design that does not use any LabVIEW motion control modules means that the client does not suffer the same operating system restriction that the server does. The client can run on Linux, or Mac OS.

The server runs on PCON, a Windows PC with a NI MXI fiber interface card in its backplane. This is the communications link to the payload. The server module is driven by a vocabulary of simple text strings. The strings represent commands at three different hierarchies of abstraction:

- low-level commands, specific to a given stepper or pneumatic
- mid-level commands, which operate at the whole-mechanism level
- high-level commands, or procedures, that execute observing sequences

This figure shows the server block diagram.
The PFIS user interface reflects this hierarchy, and uses LabVIEW tabbed controls to reveal the desired level of control at the right time.

Screen shots of the prototype GUI are shown in SALT-3140AE0023.

Interlocks

PFIS is designed with hardware interlocks sufficient to protect the instrument from damaging itself, and to prevent exposing humans to electrical shock or harm from actuators. See SALT-3140AS0015 in the PFIS CDR package. The server, however will also enforce the interlocks in software, with the goal of never actually invoking a hardware interlock.

We will implement the software interlocks at the lowest level, by "wrapping" each actuator inside a LabVIEW VI module. The VI will be responsible for check that the interlock conditions have been met before actually sending the control signal out into the hardware. If the conditions are not met, the wrapper VI will fail and pass an error cluster up the hierarchy that describes what condition was not met.

Server Modes
We define these modes for the PFIS server.

### PFIS Server Modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>All power off</td>
</tr>
<tr>
<td>Standby</td>
<td>PCON on, PXI Chassis on</td>
</tr>
<tr>
<td>Init</td>
<td>Includes sensing the instrument configuration</td>
</tr>
<tr>
<td>Ready</td>
<td>Waiting for command input</td>
</tr>
<tr>
<td>Busy</td>
<td>Processing a command</td>
</tr>
<tr>
<td>Error</td>
<td>Any detected error, including sensor error, interlock error, or actuator hangup.</td>
</tr>
</tbody>
</table>

The normal sequence of mode transitions is 1-2-3-4-5-6-7.

The Error state is entered in three different ways:

- **Transition 8**: initialization error. This usually means that PFIS was unable to sense which of its configuration states it was in.
- **Transition 9**: self-check error. PFIS will continually perform a number of self-checks in the Ready mode. The various self-checks are described in the Interlocks Specification document SALT-3140AS0015.
• Transition 10: command failure. This can happen if a reconfiguration sequence fails due to a hardware failure, or if an error is sensed from the Detector subsystem or Telescope Control System.

PFIS will spend most of its time in one of two states: Ready or Standby. PFIS will seldom go to Off because if the PXI box is turned off, then no sensor data are available during the day, and the PFIS control computer must be rebooted each time the PXI box is turned on.

State Management

In a combinatorical sense, PFIS has a huge number of possible states. It offers imaging, spectroscopy, polarimetry (3 kinds), 2 resolution regimes of Fabry Perot narrow band imaging, 3 types of CCD readout modes, and optional interactions with the telescope during an observation. The combinatoric explosion is controlled, though, by pruning whole sections of the configuration state tree according to which states are useful, and which are nonsensical.

This figure shows the useful PFIS configuration states. There are 3 main states (S1-S3), with 3 variants that select polarimetric modes (S4-S6). This figure shows the possible state transitions, and the time it takes to execute them.

PFIS States, Transitions and Readout Modes

Version 1.4, 05-Mar-2003

This state-space approach lends itself to a simple but powerful software implementation. We will control the PFIS configuration with a table-driven state machine. What is this?

PFIS State Transition Table

<table>
<thead>
<tr>
<th>Current State</th>
<th>Desired State</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 - Imaging Procedure P1</td>
<td>S2 - Spectroscopy Procedure P1, P2</td>
<td>T2: Select Filter and beam splitter for S2</td>
</tr>
<tr>
<td>S1 - Imaging Procedure P1</td>
<td>S3 - Fabry Perot Imaging Procedure P3</td>
<td>T2: Select Filter and beam splitter for S3</td>
</tr>
<tr>
<td>S1 - Imaging Procedure P1</td>
<td>S4 - Polariometric Imaging Procedure P5</td>
<td>T2: Select Filter and beam splitter for S4</td>
</tr>
<tr>
<td>S1 - Imaging Procedure P1</td>
<td>S5 - Spectropolarimetry Procedure P6</td>
<td>T2: Select Filter and beam splitter for S5</td>
</tr>
<tr>
<td>S1 - Imaging Procedure P1</td>
<td>S6 - Polariometric Fabry Perot Imaging Procedure P7</td>
<td>T2: Select Filter and beam splitter for S6</td>
</tr>
<tr>
<td>S2 - Spectroscopy Procedure P1, P2</td>
<td>S1 - Imaging Procedure P1</td>
<td>T5: Select Mask for S1</td>
</tr>
<tr>
<td>S2 - Spectroscopy Procedure P1, P2</td>
<td>S3 - Fabry Perot Imaging Procedure P3</td>
<td>T5: Select Mask for S3</td>
</tr>
<tr>
<td>S2 - Spectroscopy Procedure P1, P2</td>
<td>S5 - Spectropolarimetry Procedure P6</td>
<td>T5: Select Mask for S5</td>
</tr>
<tr>
<td>S2 - Spectroscopy Procedure P1, P2</td>
<td>S6 - Polariometric Fabry Perot Imaging Procedure P7</td>
<td>T5: Select Mask for S6</td>
</tr>
<tr>
<td>S3 - Fabry Perot Imaging Procedure P3</td>
<td>S1 - Imaging Procedure P1</td>
<td>T5: Select Mask for S1</td>
</tr>
<tr>
<td>S3 - Fabry Perot Imaging Procedure P3</td>
<td>S2 - Spectroscopy Procedure P1, P2</td>
<td>T5: Select Mask for S2</td>
</tr>
<tr>
<td>S3 - Fabry Perot Imaging Procedure P3</td>
<td>S4 - Polariometric Imaging Procedure P5</td>
<td>T5: Select Mask for S4</td>
</tr>
<tr>
<td>S3 - Fabry Perot Imaging Procedure P3</td>
<td>S5 - Spectropolarimetry Procedure P6</td>
<td>T5: Select Mask for S5</td>
</tr>
<tr>
<td>S3 - Fabry Perot Imaging Procedure P3</td>
<td>S6 - Polariometric Fabry Perot Imaging Procedure P7</td>
<td>T5: Select Mask for S6</td>
</tr>
<tr>
<td>S4 - Polariometric Imaging Procedure P5</td>
<td>S1 - Imaging Procedure P1</td>
<td>T5: Select Mask for S1</td>
</tr>
<tr>
<td>S4 - Polariometric Imaging Procedure P5</td>
<td>S2 - Spectroscopy Procedure P1, P2</td>
<td>T5: Select Mask for S2</td>
</tr>
<tr>
<td>S4 - Polariometric Imaging Procedure P5</td>
<td>S3 - Fabry Perot Imaging Procedure P3</td>
<td>T5: Select Mask for S3</td>
</tr>
<tr>
<td>S4 - Polariometric Imaging Procedure P5</td>
<td>S5 - Spectropolarimetry Procedure P6</td>
<td>T5: Select Mask for S5</td>
</tr>
<tr>
<td>S4 - Polariometric Imaging Procedure P5</td>
<td>S6 - Polariometric Fabry Perot Imaging Procedure P7</td>
<td>T5: Select Mask for S6</td>
</tr>
<tr>
<td>S5 - Spectropolarimetry Procedure P6</td>
<td>S1 - Imaging Procedure P1</td>
<td>T5: Select Mask for S1</td>
</tr>
<tr>
<td>S5 - Spectropolarimetry Procedure P6</td>
<td>S2 - Spectroscopy Procedure P1, P2</td>
<td>T5: Select Mask for S2</td>
</tr>
<tr>
<td>S5 - Spectropolarimetry Procedure P6</td>
<td>S3 - Fabry Perot Imaging Procedure P3</td>
<td>T5: Select Mask for S3</td>
</tr>
<tr>
<td>S5 - Spectropolarimetry Procedure P6</td>
<td>S4 - Polariometric Imaging Procedure P5</td>
<td>T5: Select Mask for S4</td>
</tr>
<tr>
<td>S5 - Spectropolarimetry Procedure P6</td>
<td>S6 - Polariometric Fabry Perot Imaging Procedure P7</td>
<td>T5: Select Mask for S6</td>
</tr>
</tbody>
</table>

How to use this table: the rows are indexed by the state you are currently in. The columns are indexed by the final (not next) state that you want to end up in. For each (current,desired) state pair, the indexed table cell tells which transition to execute next, and what state that takes you to. Keep iterating until you end up in the desired state (you are done when you end up on the diagonal).
Example: Suppose you are in state S5 (Spectropolarimetry), and you want to be in state S3 (Fabry-Perot Imaging) for the next observation.

1. (S5,S3) indexes the cell (-T2, S2). So execute transition -T2 (Remove waveplates). You are now in state S2 (Spectroscopy).
2. (S2,S3) indexes the cell (-T1, S1). So execute transition -T1 (Articulate camera back to home). You are now in state S1 (Imaging).
3. (S1,S3) indexes the cell (+T3, S3). So execute transition T3 (Insert etalons). You are now in state S3.
4. (S3,S3) indexes the diagonal. You are done.

This table-driven approach has many advantages over a procedural approach. For example, in the example given above, one made the move from S5 to S3 by:

- de-articulating the camera arm to its home position
- inserting the Fabry-Perot etalons.
- removing the waveplates and polarizing beamsplitter

If, for some operational reason, one wanted a different order, say:

- removing the waveplates and polarizing beamsplitter
- de-articulating the camera arm to its home position
- inserting the Fabry-Perot etalons.

then one has merely to update the state table. No procedures have to be written or modified. This approach has been used before at the Space Astronomy Laboratory, in Jeff Percival's amazing astrometric kernal, the "Telescope Pointing Machine" (TPM). The state diagram and detailed documentation are available on the web.

The PFIS configuration state table also represents a higher, 3rd level of interlock protection. The state table entries ensure that the instrument will be moved through a tested and well-known sequence of states. The hardware interlocks are shadowed by the low-level LabVIEW software interlocks, ensuring that the hardware interlocks will not be tripped. Likewise, the low-level interlocks are shadowed by the state table entries, ensuring that the software interlocks will not be tripped!

One might complain that using a table-driven state machine precludes the possibility of saving time by operating mechanisms in parallel. We point out, though, that the two longest configurations, the etalons and the camera articulation, are mutually exclusive in a mechanical sense, and therefore cannot be done in parallel. Moreover, the three reconfigurations that don't result in a state change (slit masks, gratings, and filters) can be done in parallel, and we will use LabVIEW's parallel execution capabilities to do so.

**Observing Procedures**

PFIS commands are organized into a 3-level hierarchy, allowing access at the actuator, mechanism, or procedure level. We want to avoid controlling PFIS observing sequences by sending large numbers of
low-level commands from the client (user interface) to the server. The knowledge of how to execute an observing sequence should lie in the server.

We have identified the observation sequences that make sense for each of the configuration states. These are discussed in the Operational Concepts Definition Document (SALT-3170AE0002).

Error Management

We will manage errors using the standard error management tools in LabVIEW. Each VI will take an error input and provide an error output. The error flow will be wired up, one VI after another, in LabVIEW fashion. Each VI will inspect its error input, and decline to act if an error is indicated. If action is declined, then the input error will be passed on. If action is taken and an error is produced, then that error cluster will be passed on.

The error cluster will contain a specific description of the error: the interlock code, the sensor responsible for the failure, and so on.

Combined with a hierarchical set of VIs, this method will allow errors to flow up to the top, then over to the client interface.

Data and Command Management

All PFIS commands (text strings) and data clusters will be defined as LabVIEW "type def" clusters. The typedef clusters will all live in a single directory, allowing a convenient way of inspecting or using them en masse.

Communication with Other SALT Subsystems

PFIS must communicate with two other SALT subsystems: the SAAO CCD subsystem (PDET) and the Telescope Control System (TCS).

It sends set-up commands to PDET for configuring the CCD for exposures, and trades information with PDET for handshaking, slit mask peakup exposures, and FITS header keywords. No raw image data are exchanged; the interactions will be conducted using text strings. The vocabulary will be built up during the fabrication phase by the SAAO and PFIS teams.

PFIS interacts with the TCS to do slit mask peakups, and for the shuffle-and-nod observing procedures. These interactions will also use text strings. The vocabulary will be built up during the fabrication phase by the TCS and PFIS teams.