Southern Africa Large Telescope

Prime Focus Imaging Spectrograph

SAAO Detector Subsystem:

SALT-3199AS0001: Software Development Plan

SAAO PFIS Detector Subsystem Team:
Dave Carter
Luis Balona
Etienne Bauermeister
Geoff Evans
Willie Koorts
James O'Connor
Darragh O'Donoghue
Faranah Osman
Stan van der Merwe

Issue 1.3
25 June 2003

Issue History

Number And File Name	Person	Issue	Date	Change History
SALT-3199AS0001 Software Issue 1.1.doc		1.1	11 Feb 2003	PFIS CDR version
SALT-3199AS0001 Software Issue 1.2.doc		1.2	19 Mar 2003	Post-PFIS CDR version from discussions at Wisconsin
SALT-3199AS0001 Software Development Plan Issue 1.3.doc		1.3	25 Jun 2003	Design phase: split of CDR software doc into development plan document and software specification and design document. Both are designated to be Issue 1.3.

TABLE OF CONTENTS

1 Scope 4
2 Referenced Documents 4
3 Description of Software To Be Developed 4
3.1 Requirements Analysis 5
3.2 Software Specification Review (PDR) 5
3.3 Software Design 5
3.4 Software Critical Design Review (CDR) 6
3.5 Software Coding and Debug 6
3.6 Software Code Reviews 6
3.7 Module Testing 6
3.8 Software Testing 6
3.9 Subsystem Commissioning and Integration 6
3.10 Software handover 6
4 Software Safety 7
4.1 Safety certificate 7
4.2 Communication Integrity 7
4.3 Initialisation 7
4.4 Start-up and Shut Down Procedure 7
5 Generic Software Requirements 7
5.1 Naming and Tagging Conventions 7
5.2 Remote Initialisation 8
5.3 Data 8
5.4 Software cyclic execution 8
5.5 Data time stamping 8
5.6 Modular Design 8
5.7 Measuring Units 8
5.8 Synchronisation 9
5.9 Unused Code 9
5.10 Software Comments 9
5.11 Self-changing code 9
5.12 Communication methods 9
6 Specific PDET Software Requirements 9
6.1 Operating Systems 9
6.2 Development Software 9
6.3 Application Software 10
6.4 Man-Machine Interfaces 10
7 Deliverables 10
8 Configuration Control 10

ACRONYMS AND ABBREVIATIONS

ATP	Acceptance Test Procedure
ATR	Acceptance Test Report
BITE	Built-in Test Equipment
CDR	Critical Design Review
COTS	Commercial off the shelf
ELS	Event Logger Software
HET	Hobby-Eberly Telescope
I/O	Input/Output (Device)
ICD	Interface Control Dossier
MMI	Man-Machine Interface
MTBF	Mean Time Between Failures
MTTR	Mean Time to Repair
OEM	Original Equipment Manufacturer
OPT	Operational Planning Tool
PC	Personal Computer
PDR	Preliminary Design Review
PFIS	Prime Focus Imaging Spectrograph
PI	Principal Investigator (Astronomer)
PIPT	PI Planning Tool
PLC	Programmable-Logic Controller
SA	SALT Astronomer
SALT	Southern African Large Telescope
SAMMI	SA Machine Interface
SC	Software Component (e.g. part fo the TCSS)
SDB	Science Database
SD	Software Design
SDP	Software Development Plan
SI	Software Item (the TCSS is a Software Item) *OR*
SO	SALT Operator
SOMMI	SO Machine Interface
SRS	Software Requirement Specification
STARCAT	Object Catalogue
SW	Software
TBC	To Be Confirmed
TBD	To Be Determined
TCS	Telescope Control System
TCSS	TCS Server
VI	Virtual Instrument (Labview function) *OR*

1 Scope

The PFIS detector package is being supplied to the University of Wisconsin by the SAAO. This document specifies the software development plan for the Detector Package.

2 Referenced Documents

The following documents are referenced in this specification and are applicable to the extent specified herein.

1000AA0030	SALT Safety Analysis
1000AB0044	SALT Labview Coding Standard
1000AD0005	SALT Computer Architecture
1000AS0040	SALT Operational Requirements
1700AS0001	TCS Specification
1773AS0001	TCS Interlock Panel Specification
1000AS0049	SALT Data Interface Control Dossier

3 Description of Software To Be Developed

The PFIS detector software comprises the following computers and units/applications. Only the software in **bold** is new application software that is covered by the development plan. The software in ***bold italics*** is assumed to be the responsibility of the University of Wisconsin or the SALT Project.

a. PFIS Control PC (hereafter called PCON)
   - ***PCON Control Program Software.***
   - ***PCON Detector Interface (designated PCONDI) Software. This is the main interface to the PDET.***
   - Labview Data Socket (part of the standard Labview Application)

b. PFIS Detector PC (hereafter called PDET)
   - **PDET Kernel Software (designated PDET KER). This will interact with the PCONDI residing in the PCON machine. PDET KER will also interact with PDET MMI and PDET CON as described below.**
   - **PDET MMI Software (designated PDET MMI). This is the interface to PDET for development and maintenance via the PDET PC keyboard. It will be similar to PCONDI, although the latter will exert control not via an MMI.**
   - **PDET Control Software (designated PDET CON). This is software that will receive instructions from PDET KER, and control all the detector hardware.**
   - **PDET PCI Card Software (designated PDET PCI). This is software that is supplied by Astronomical Research Cameras with their SDSU II/III CCD controllers.**
   - **PDET SDSU II/III Control Software (designated PDET SDSU), including the software in the subsytem controller. This is software that is initially supplied by Astronomical Research Cameras with their SDSU II/III CCD controllers. The supplied software will be used as a prototype for an SAAO developed equivalent, tailored for the PDET application.**
   - *This machine will contain no other applications.*

In addition, the PDET software will interact with these machines forming part of the SALT TCS. Their main applications software units are also indicated:

c. Data Processor
   - Science database. This is the organised storage and retrieval of all instrument configuration, calibration, science and telescope data pertaining to science observations made.

d. Data Reduction PC
   - **PDET Data reduction pipeline (designated PDET DRED).**

e. TCS Server

- SALT TCS Server application
- Labview Data Socket (part of the standard Labview Application)

Table 1 shows the planned schedule for the work.

Table 1: PDET Software Schedule

Milestone	Date
Development Plan & Specification	March 2003
Design & Prototype	15 Jun 2003
Coding & Test of Modules Complete	15 Sep 2003
Integration Complete	15 Oct 2003
ATP	Mid-Oct 2003
Science data reduction pipeline	1 Mar 2004

3.1 Requirements Analysis

A distinct software specification (SRS) shall be written for each of the following software items, using the document number 3199AS0001:

Title	Designation
PDET KER Software Specification	PDET KER
PDET MMI Software Specification	PDET MMI
PDET Control Software Specification	PDET CON
PDET PCI Card Software Specification	PDET PCI
PDET SDSU Control Software Specification	PDET SDSU
PDET Data Reduction Software Specification	PDET DRED

We assume there already exist requirements analysis for PCONDI.

3.2 Software Specification Review (PDR)

Either prior to starting the software design, or after the design is complete (at the latest), the Software Requirement Specification shall be reviewed by the development team, the University of Wisconsin, and the SALT Team. The purpose of the review is to verify that the software requirements have been correctly identified.

3.3 Software Design

Prior to coding the software, it is essential to structure and design the software to meet not only the functional requirements of the SRS, but also the maintainability, reliability and testability requirements. The output of the Software Design process will be in various forms, but the major design aspects shall be documented in the *Software Design Section (SD)*. At least the following shall be addressed:

a. A high-level **design description**, describing the overall integration and interaction of the modules how this relates to the software states and modes.
b. An updated copy of the **software architecture diagram**.
c. A detailed **functional-flow and data-flow diagram**, showing all the software modules and the precise data flowing between them. The implementation of specific timing, synchronisation and interrupts requirements shall be illustrated. For Labview software, the mechanism of data flow (i.e. wires or VI server calls) shall be identified.
d. The **software design** of each module must be provided. This shall indicate the specific data inputs, outputs, processing and timing requirements for that module and shall give specific formula's and algorithms that are to be executed. Details of global variables, interrupt operation, timing implementation and shall be defined. The design shall be documented in pseudo-code, flow diagrams or English narrative.

3.4 Software Critical Design Review (CDR)

Prior to full-scale software coding, the software design shall be reviewed by the supplier development team and the client. The purpose of the review is to verify that the requirements of the SRS and other implicit requirements have been adequately and efficiently addressed in the design. It is an opportunity for the development team to co-ordinate the hardware, software and equipment designs and to ensure that non-functional requirements such as maintainability, testability and reliability are adequate.

The CDR shall address the overall software design (architecture, data flow, timing) and detailed design of each module.

3.5 Software Coding and Debug

During this process the software code for each module is generated according to the design defined in the SD. Specific coding standards, metrics and conventions are applied (as defined elsewhere in this document) and software comments inserted.

In parallel with the software coding process, a *Software Acceptance Test Procedure (ATP)* is defined and documented by the developer. Tests shall be defined to verify that the software complies with each requirement of the SRS. This document shall subject to approval by the University of Wisconsin and the SALT Project team.

3.6 Software Code Reviews

The source code of each completed module is reviewed by the development team to check the appropriateness of software style, efficiency and to co-ordinate interfacing modules. The appropriate method of testing each module shall be agreed. The client may at his discretion attend such reviews. A record shall be kept of each review and the comments recorded. The implementation of such comments shall be verified during module testing.

3.7 Module Testing

Software modules shall be individually tested prior to integration with the other modules, to an appropriate level. Testing a module may use either a simple stub simulating interfaces to other modules or another module (or group of modules) that has already been tested. The results of each module test shall be recorded, albeit informally.

3.8 Software Testing

Tested modules are incrementally integrated together and progressively checked. When all the software has been integrated, the tests defined in the Software ATP shall be executed where possible. The precise hardware and software configuration tested shall be defined. From this point forward, all software changes shall be logged. A TCS Server simulator shall be used to verify the communications interface to each computer *prior to delivery of that computer and its software by the developer*. A communication test using the simulator shall be part of each computer item's ATP.

At this point the software shall be fully under Configuration Control (see section 8) and all software changes managed.

3.9 Subsystem Commissioning and Integration

The next step of the process is to integrate the software with the PCONDI software. Commissioning is complete when the PDET Software ATP, which verifies the performance against its specification, has been passed.

The final step of the process, during which the final aspects of the software items performance is verified, is the System Integration, when all the subsystems are integrated to form an operating instrument. Only when the PDET ATP has been successfully completed, can each SW item be said to be complete.

3.10 Software handover

During step 3.9, the responsibility for maintenance of the software is transferred from the original developer to the U. of Wisconsin. This delivered software package shall contain a full definition of the latest software configuration, including the following:

a. A *Version Definition* – a table indicating the current revision numbers of each of the software modules of each software item
b. The *Software Configuration Definition* – an electronic copy of all configuration data for operating systems, firmware, set-up data, calibration constants, user-defined parameters etc.
c. The *Software Source Code* of the present software version
d. *Original legal copies* of the operating systems, compilers, tools, utilities that are required to maintain the software
e. Final copies of *Operating, Maintenance and Calibration procedures* where applicable
f. A final version of the *Safety Certificate*

4 Software Safety

4.1 Safety certificate

A Safety Certificate shall be issued for PDET Software. The certificate shall identify all the software items that form part of the PDET software suite.

4.2 Communication Integrity

Communication integrity between subsystems and all equipment items shall be monitored by all items receiving data. Failure to receive correct data or failure to receive any data from a particular device shall be reported the operator via the Event Logger.

Detection of communication failure shall be facilitated by using the "Validity Word" in the communicated Labview data, or a similar method for non-Labview Software.

Each software item shall fail in a safe fashion if it does not receive the required data. Gradual degradation of system performance should be allowed where possible.

4.3 Initialisation

PDET software shall be in a safe state when un-initialised or switched off. Similarly, un-initialised inputs (e.g. from other subsystems) shall not cause incorrect responses from the software.

The following initialisation sequence shall be followed by all software:

a. Switch all outputs to a safe state (e.g. motors, OFF)
b. Indicate "Initialisation State" to the operator
c. Check the integrity of the processing hardware and memory using simple arithmetic checks
d. Check communication with and correctness of peripheral devices (if applicable)
e. Verify the correctness of configuration data and then initialise variables accordingly
f. Check communication with interfacing computers
g. If all operations are successful, report "System Okay" to the operator and enter into a "ready" state, where after the state will be determined by switches, commands, data etc. If operations a. to e. are not successful, report "System Start Failure" and indicate the type of failure encountered. If communications with another computer cannot be established, this should be reported.

4.4 Start-up and Shut Down Procedure

During Start-up and Shut Down, preventative measures shall be taken to handle process conditions as well as Inputs and Outputs in a safe manner.

5 Generic Software Requirements

5.1 Naming and Tagging Conventions

Each SW component shall be uniquely identified with a sensible name. File extensions native to the programming language used, shall be adhered to (i.e. Labview files *.vi, *.glb, *.ctl and *.rtm)

All variables, memory and block naming shall be clear, logical and understandable. A uniform convention shall be used throughout an item, preferably using whole English words. Where compilers/interpreters do not support long variable names, a consistent abbreviation may be used, with a clear definition in the appropriate documentation.

Naming conventions will be agreed during the Software PDR.

5.2 Remote Initialisation

It shall be possible to trigger the initialisation sequence described in 4.3 remotely via the normal communication to an item. (e.g. The operator must be able to send a "reset" command across the Ethernet to any computer to trigger initialisation). This is not applicable to MMI applications.

5.3 Data

A set of Critical Data, over and above data required for functional operation, shall be agreed with the client for each computer item. This data set shall be updated at a rate of at least 1Hz and be sent to the Event Logger:

- Item Mode
- Item Health Status
- Fault list

This Data Set will be finalised during the Critical Design Review.

Where internal variables may assist diagnostics, their values should also be transmitted to the Event Logger.

5.4 Software cyclic execution

After the completion of initialisation, the code of a SW item shall execute in a cyclic fashion, at a constant rate, commensurate with the control bandwidth/frequency/latency required.

5.5 Data time stamping

Time-critical data will be agreed with each supplier and identified as such in the ICD. All such data shall be time-stamped in an agreed fashion to facilitate synchronisation of subsystems.

5.6 Modular Design

Software shall be designed in a scalable and modular fashion. All software modules (i.e. Labview VI's, procedure and functions – see SALT Labview Coding Standard) shall be designed to minimise their data interfaces and to group functions that belong together, keeping in mind future growth and hardware upgrades. Compliance to these requirements shall be demonstrated at the PDR, CDR and code reviews. In particular, the following types of functions shall be in independent modules:

- Input/Output hardware communication drivers
- Input/Output scaling from hardware units (e.g. 1024bits) to/from engineering units.
- Initialisation sequences
- User configuration sequences
- Equipment mode/state control
- Mathematical/control algorithms
- Data storage and retrieval
- Data communication
- Fault monitoring and reporting

Identical software functions shall not be repeated in different areas but rather grouped together as a shared function or procedure.

5.7 Measuring Units

The SI metric system shall be used for all processing except for angles, which shall be in Radians. The units of information displayed on MMI displays shall be in "human-friendly" units and will be agreed during the CDR.

5.8 Synchronisation

Two methods of synchronisation are allowed, the selection of which shall be commensurate with the time accuracy requirements and shall be subject to approval during the PDR.

a. *Network synchronisation*: An NTP server will provide accurate GPS time to all subsystems requesting this via Ethernet. The accuracy of this time should be better than 150ms and would be suitable for most applications.
b. *Hardware synchronisation*: A precision hardware time signal (e.g. 1 pulse-per-second and 10MHz) will be made available to all items requiring very accurate time (e.g., Tracker Computer, Payload Computer and Instrument Computers). A computer input reads this signal and synchronises SW functions accordingly.

5.9 Unused Code

All unused code and variables shall be removed from the software.

5.10 Software Comments

Over and above the software development documentation, the following documentation shall form and integral part of the software code in the form of comments or function help:

- Each software module shall have a header or associated "help" definition describing the following:
  - The name and purpose of the module
  - The inputs and outputs of the module and their types
  - A detailed description of the functions performed by the module. (This may be English narrative or pseudo-code).
- A definition/description of local and global variables used in the module
- English description of the actions performed by SW code. As a guideline give one line of comments per two lines of code for text-based software. For Labview software each VI shall have help information defined, as indicated in the first bullet.

5.11 Self-changing code

Self-changing code shall not be allowed.

5.12 Communication methods

Communication between computers on the Ethernet network, shall be TCP/IP-based Data socket communication or http://, as agreed.

A central database (ICD) shall identify the source, destination, type and update frequency of each parameter. Communications software for Labview items will be provided by SALT, according to the required data types.

6 Specific PDET Software Requirements

6.1 Operating Systems

The Linux or possibly the Real Time Linux Operating System is proposed for the PDET PC.

6.2 Development Software

The following development environments are proposed:

- PDET KER: Labview 6
- PDET MMI: Labview 6
- PDET CON: C

- PDET PCI: C
- PDET SDSU: DSP Assembler
- PDET DRED: SQL, C

6.3 Application Software

The SALT Labview Coding Standard shall be applied to all Labview software. The Software Engineering Primer in "Labview Power Programming" and "Internet Applications in Labview" are recommended reading.

6.4 Man-Machine Interfaces

All MMI's shall be subject to approval by the University of Wisconsin.

7 Deliverables

Unless agreed otherwise, at least the following items shall be delivered for each software item.

a. Original documentation and electronic media of all the bought-out software installed on the computer (including operating systems and device drivers).
b. Original documentation and electronic media of the software development environment in which the software was developed (e.g. compilers, version control tools etc.), unless agreed otherwise.
c. Development environment and operating system configuration data (e.g. memory map set-ups, compiler directives, copy of Linux configuration files, Windows INI files, registry files) on CD-ROM.
d. Documentation as described in section 4 (if applicable according to Table 1), including the following:
    - Software Requirement Specification
    - Software Design Document
    - Software Code Review report/minutes
    - Software Acceptance Test Procedure
    - Software Acceptance Test Report
e. All application software source code, compiled software, installation software and configuration information on CD-ROM.
f. Calibration, Maintenance and operating procedures if applicable
g. A Version Definition (See section 8)

The following will be available for the PDET Software as a whole:

h. A final version of the Safety Certificate
i. A Software Development Plan (this document)
j. Acceptance Test Procedure and Report

8 Configuration Control

Each SW Item shall be uniquely identified by the PFIS configuration name defined earlier.

During the software coding, testing and integration process, maintain the Labview version control tool shall be used, whereby each software module has a **revision** number reflecting changes made. Changes after initial SW integration shall be controlled and documented. Every update to that module shall result in a change in the revision number of that module. Modules of previous revisions shall not be overwritten or destroyed but kept for recovery purposes.

The integrated software comprising many modules (i.e. the SW Item), shall also have a unique **version** number at critical stages in the process (e.g. at Software Testing, delivery etc.). A document (the *Version Definition*) shall be maintained which records the included modules and their revision numbers making up each version of a SW Item.

The following numbering scheme is preferred for revisions and versions:

Example: Tracker Payload SW Version 2.32 comprises the following modules
- Module A: *Initialisation* Revision 5

- Module B: *Hardware set-up* Revision 31
- Module C: *Mode control* Revision 29
- ……

Where the number before the dot is incremented with major changes to the software or to mark a significant event (e.g. software delivery) and the number after the digit changes with minor modifications.