Computer Applied Techniques Ltd
Malahide
Co. Dublin Tel : +353.1.8450921
IRELAND Fax : +353.1.8450136

SPIRE-REF-REP-002099

HERSCHEL-PLANCK

Instrument Software

Support

Status Inventory Report for SPIRE ICU SW

Doc Ref H-P-1-CAP-RP-0007
Issue 1 revision -
Date 11th. June, 2004

PTI : < > PC: < 1 > DRL: < > Model : < >

Author	:	E.L. Bach	
Checked by	:	R.D. Cluff/G. Duncan	
Agreed by	:	E.L. Bach	
Authorised by	:	F.J. Kennedy	

Controlled Copy No. .....

DISTRIBUTION SHEET

NAME	COMPANY	DATE	COPY NO.
Internal			
Master File	CAPTEC	11/06/04	1
Technical File	CAPTEC	11/06/04	2
E.L. Bach	CAPTEC	11/06/04	3
R.D. Cluff	CAPTEC	11/06/04	4
F.J. Kennedy	CAPTEC	11/06/04	5
External			
A. Elfving	ESA	11/06/04	6, 7, 8
K. J. King	RAL	11/06/04	9

DOCUMENT CHANGE RECORD

DOCUMENT TITLE			
Status Inventory Report for SPIRE ICU SW			
DOCUMENT NUMBER	ISSUE	REV.	DATE
H-P-1-CAP-RP-0007	1	-	11/06/04

PAGE	PARAGRAPH	REASON FOR CHANGE
All	All	New Issue

DOCUMENT STATUS SHEET			
DOCUMENT NUMBER H-P-1-CAP-RP-0007		DOCUMENT TITLE Status Inventory Report for SPIRE ICU SW	
ISSUE	REVISION	DATE	REASON FOR CHANGE
1	-	11/06/04	First issue

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose

This document is the Report for the Status Inventory of the SPIRE ICU software. It is an input to the recovery plan for the SPIRE ICU software.

1.2 Scope

This document applies to the state of software documentation, code, testing and planning as determined from documents submitted to ESA at the time of audit performance and submissions made in answer to questionnaires or at the audit at the software developer premises. This information is provided or referenced within this status inventory report.

Actual recommendations for recovery are beyond the scope of this report. Instead this report will provide:

- identification of the needs of the software development, prioritised according to corresponding project risks;
- assessment of completion status and projected status as provided by the software developer;
- assessment of quality of provided documentation and code;
- identification of missing documentation deemed necessary for a minimum document set.

1.3 Background

The IHDR for the SPIRE instrument (ref. /6/) identified problems with assessing the state of software development and planning. As similar issues were present to a lesser or greater degree in other instrument software developments, ESA decided to inaugurate a programme of instrument software support with the following objectives (see /5/):

- To assess current software status per software item and software team;
- To review the associated software development documentation, paying particular attention to and providing expertise in planning, configuration and operations/maintenance issues, in the production of quality documentation and in the application of ESA standards;
- To construct a synthesised status report identifying the areas where specific attention is needed to overcome inadequacies or to ensure successful completion of software development;
- To construct recovery plans, specifically tailored for individual software items and software teams, defining the required actions pertaining to development, documentation, configuration control, and verification, and supplying a schedule for these actions.

The status inventory is determined for each of the different instruments on the Herschel and Planck mission. The current document is for the SPIRE instrument.

1.4 Document Overview

The following section contains the references for the audit. Section 3 is a record of the audit. Section 4 contains the audit conclusions, together with summary lists of software needs, required document updates and lessons learned.

1.5 Acronyms

ADD	Architectural Design Document
ASI	Agenzia Spaziale Italiana
AVM	Avionic Model
CAPTEC	Computer Applied Techniques, Ltd.
CDMS	Command & Data Management System
CDR	Critical Design Review
CG	Carlo Gavazzi Space
CGS	Carlo Gavazzi Space
CM	Configuration Management
CPU	Central Processing Unit

CRC	Cyclic Redundancy Check
CVS	Concurrent Versions System
DDD	Detailed Design Document
DPU	Digital Processing Unit
DRCU	Detector Readout & Control Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
ESA	European Space Agency
FDIR	Failure Detection, Isolation & Recovery
FM	Flight Model
H-P	Herschel-Planck
HIFI	Heterodyne Instrument for Herschel
HK	Housekeeping
HW	Hardware
I/F	Interface
ICC	Instrument Control Centre
ICD	Interface Control Document
ICU	Instrument Control Unit
ID	Identity
IFSI	Istituto di Fisica dello Spazio Interplanetario
IHDR	Instrument Hardware Design Review
ILT	Instrument Level Test
ISO	Infra-red Space Observatory
ISS	Instrument Software Support
LWS	Long Wave Spectrometer
MRB	Materials Review Board
N/A	Not Applicable
NCR	Non-Conformance Report
OBS	On-Board Software
PA	Product Assurance
PACS	Photoconductor Array Camera & Spectrometer
QM	Qualification Model
RAL	Rutherford Appleton Laboratory
SCIDL	Software Configuration Item Data List
SCR	Software Change Request
SPIRE	Spectral & Photometric Imaging REceiver
SPMP	Software Project Management Plan
SPR	Software Problem Report
SPU	Signal Processing Unit
SRD	Software Requirements Document
SRON	Space Research Organisation Netherlands
SSD	Software Specification Document
SUM	Software User Manual
SVVP	Software Verification & Validation Plan
SW	Software
TBC	To Be Confirmed
TBD	To Be Defined
TC	Telecommand
TM	Telemetry

URD	User Requirement Document
VM	Virtual Machine

2 SERVICE INFORMATION

2.1 Audited Material

/1/ "SPIRE OBS User Requirements Document", SPIRE-IFS-PRJ-000444, Issue 1.2, 15/05/03

/2/ "SPIRE OBS Software Specification Document", SPIRE-IFS-PRJ-001036, Issue 1, 18/05/03

/3/ "SPIRE OBS Verification and Validation/Acceptance Test Plan", SPIRE-IFS-DOC-001392, Issue 1.2, 24/02/03

/4/ "SPIRE OBS Software User Manual", SPIRE-IFS-PRJ-001391, Draft 0.7, 15/09/03

2.2 Additional References

/5/ "Support Plan", 584-011, Issue 1A, 08/03/04

/6/ "SPIRE Instrument Hardware Design Review Board Report", SCI-PT/19056, Issue 1, 22/09/03

/7/ "Operations Interface Requirements Document, SCI-PT-RS-07360, Issue 2.2, 31/09/03

/8/ "Space/Ground Interface Control Document", SCI-PT-ICD-07418, Issue 3.1, 01/12/03

/9/ "Packet Structure Interface Control Document", SCI-PT-ICD-7527, Issue 4, 07/10/03

/10/ "Space Project Management: Risk Management", ECSS-M-00-03A, 25/04/00

2.3 Record of Meetings

The following meetings took place for this audit:

Meeting	Location	Date	Participants
IFSI Code Audit	IFSI, Rome	28/05/04	S. Molinari, G. Liu (IFSI SPIRE) S. Pezzuto (IFSI PACS) C. Scharmberg (ESA) K. King (RAL) E. Bach (CAPTEC)

3 AUDIT DEFINITION AND RESULTS

3.1 Follow-up of Previous Audits

No previous audits performed.

3.2 Scope of Present Audit

3.2.1 Documentation

The assessment of documentation was performed on ref. /1/-/4/ using the checklists from ref. /5/. Detailed ratings are given in §4.3.

The documents assessed were the URD (/1/), the SSD (/2/), the SVVP (/3/) and the SUM (/4/).

Specific questions were added to the audit questionnaire for response by the software developer.

It was not felt necessary to present details of the application of the checklists; the results have been issued as specific questions in the questionnaire supplied to the software developers (see Appendix A).

3.2.2 Code

Code was provided by the software developer. Some comments (see figure 3-1) were supplied at the audit and handled during the meeting. This code was baselined as ILT revision 1.2i but was stated not to be a formal release. From the samples of code studied it is clear that configuration management procedures are working and that the code maps well to the architecture, although commenting should be improved.

3.2.3 Questionnaires

Detailed questionnaires (see Appendix A) were sent to the instrument focal point and to the software developers, and were completed. The responses were discussed at the meeting.

1. The external variables HIgh_Prio is not used in write_ee.c and HIgh_Prio, etc. are never used in file eprm.c, because the lines referencing them are comment lines.

   A: Clean-up needed on CG software.

2. The use of the files all_include.h, all_ext.h and all_xfg.h is not recommended, because it does not hide external variables from functions which do not need them. This also complicates semi-automatic analysis of data shared between tasks.

   A: Will be considered when software is stable.

3. The external variables hRet, etc. are never used in tabler.c, because the line referencing them is a comment line.

   A: Will be considered when software is stable.

4. The variable Task_index is referenced by eprm.c in an extern statement but never used.

   A: Clean-up needed on CG software

5. The variable tblStackPtr is used by hk_monitor, tabler and cmd_seq tasks. What is the consequence if the following line (not one assembler instruction!) in tabler is interrupted by one of the higher priority tasks?

   tblStackPtr += sizeof(struct tblSquare) * Max_tableId_numbers + NO_tbl_leap;

   Can the interrupting task use a corrupt value of the variable?

   A: Prevented by tblStatus flag.

6. The variable TC__packet is accessed by the VM task and the cmd_seq task, which performs multiple writes to the variable. How is it ensured that the VM task does not enqueue a corrupt value, i.e., if it executes between the cmd_seq writes to the variable TC__packet?

   A: IFSI to analyse and document constraint and update code if necessary.

7. The use of malloc in eprm.c should be avoided and the allocation made statically.

   A: IFSI to raise with CG

8. The function MilMalloc in Mildef.h is never used.

   A: Clean-up needed on CG software.

9. Why is nalloc used and how is it ensured that an interrupted process will not access invalid memory if it is interrupted by a process that calls nalloc?

   A: Use of critical section clarified by IFSI. SSD description will be improved.

10. Are you able to state which variables are accessed from more than one task? This is not easily discernible given the coupling of code by inclusion of global headers.

    A: Use of critical section clarified by IFSI. See point 2.

11. Note code does not enter main while loop in generate_dump packets if ground requests a 0-length dump. Does this have any consequences?

    A: IFSI to analyse effect of 0-length dump/CRC commands.

12. Note code contains comments in Italian and is not consistently commented as to purpose and function.

    A: IFSI to consider and update where necessary—this is normal work.

Figure 3-1 Code comments presented at meeting

4 CONCLUSIONS & ANALYSIS

4.1 General Conclusions

The audit was felt to have worked well with good co-operation from all participating parties. Certain needs have been identified as critical in order to maintain the software development schedule; these are noted in §4.4.1.

4.2 Software Schedule and Completion Status

4.2.1 Software Schedule Assessment

Dates were given as follows:

AVM end July 2004 (TBC) with final delivery September 2004;
FM November 2004.

These dates are compatible with instrument and ESA needs. However, particular problems are identified with the achievement of the schedule; associated needs are presented in §4.4.1.

4.2.2 Software Completeness Assessment

The breakdown given below is based on the responses of the software developers to the questionnaires.

Task	Completion status
Requirements analysis	90% complete
Architectural design	80% complete
Detailed design	N/A, according to the OBS PA plan
Coding	75% complete
HW/SW integration testing	90% complete
Verification	25% complete
Operations and maintenance planning	No planning at all, except for a general agreement about the OBS maintenance responsibility and implementation scenario.
Documentation of the tasks results	25% complete

These figures are as reported by the software developer and require some clarification:

1. IFSI understanding of the baseline needs to be confirmed (there is an agreed redline, but no formal issue by RAL).
2. Completion of documentation varies; the architecture and verification are less complete than the software requirements and the user manual, which are considered by the software developer to be more essential for development.

4.3 Detailed Ratings

An ordinal scale of 0-4 is used to rate audited documents corresponding to the list in /5/, i.e.,

0 document not supplied for assessment
1 document does not fulfil stated purpose
2 document contains major open areas and/or major errors
3 document contains minor open areas and/or minor errors
4 document is acceptable as is

An ordinal scale of 0-4 is used to rate questionnaire responses, i.e.,

0 concern not addressed
1 concern addressed inadequately
2 concern addressed adequately but not consistently
3 concern addressed consistently but not fully
4 concern addressed fully

The resulting ratings for documentation are given in the following table under the heading "Status: Docs". The ratings for questionnaire responses are given e.g., as Q1 in the following table, which groups the questions

according to what aspect of development the question is concerned with and provides a mnemonic for the content of the question.

Status: Docs	SPIRE ICU	Justification
URD	2	The URD does not specify FDIR requirements or pointing algorithms.
ICD TM/TC	0	A RAL document has been provided but is outside the scope of this audit.
ICD HW/SW	0	
SRD	2	The SSD does not specify FDIR requirements or define algorithms.
ADD	2	The SSD architecture does not define the interface between objects or identify interface data types or operations that are part of the interface.
DDD	0	
SUM	0	
SVVP	2	Test definition is not documented for software tests and there is no verification matrix.
SCIDL	0	
SPMP	0	A RAL document has been provided but is outside the scope of this audit.
Status: Baseline		
Q1 baseline chg	3	From discussion at the meeting it appears that no areas remain except for FDIR, command list entries to be specified by RAL, and formalising the baseline—both IFSI and RAL were satisfied a freeze would not pose a problem.
Q2 req verif	2	Test definition is not documented for software tests and there is no verification matrix.
Q3 mem/CPU use	4	Peak stressed CPU load measured at less than 20%.
Q4 software CM	3	Configuration management seems to be in place and working, but IFSI does not always document bugs as SPR.
Status: Team/Tools		
Q5 team makeup	1	Manpower allocation admitted to be inadequate. Experienced personnel are not taking on a management role.
Q6 uniform dev	4	Uniformity achieved through single developer.
Q7 previous exp	2	Experienced personnel are not taking on a management role.
Q8 software tools	3	Tools seem adequate, except in verification. Use of Splint is also recommended as a tool for use in code inspections.
Status: Completeness		
Q9 task complete	2	Planned completion date is inconsistent with current completion status and manpower allocation.
Q10 complete dates	3	Baseline freeze is necessary and late hardware delivery is also a concern.
Q11 open areas	3	Open areas are well understood or outside IFSI control.
Q12 metrics	0	No metrics are used.
Status: I/F		
Q13 i/f with others	4	Interfaces are adequate for inputs. Late hardware delivery is covered elsewhere.
Q14 i/f control	4	Technical interfaces are adequate.
Q15 review/monit	3	Instrument team monitoring mechanisms are sufficient, although they do not always ensure early visibility of problems; there are minor problems of mismatch in expectations between IFSI and SRON regarding reporting.
Supplier Name	IFSI	
Supplier Locn	Rome	

4.4 Real and Potential Problems

4.4.1 Needs to be Considered in Recovery Plan

Regarding interactions with hardware

SP-IF-N-1 Boot software and drivers need to be verified against requirements and thoroughly tested at unit level.

SP-IF-N-2 Software acceptance test definition needs to be checked. In this regard:

- Realistic command lists need to be tested using the actual hardware, to ensure command list mechanism is adequate for operations;
- Suitability of acceptance test definition for RAL test environment needs to be confirmed.

SP-IF-N-3 Before it is used in tests at Astrium, software needs to be adequately tested at a lower level.

SP-IF-N-4 DPU hardware needs to be available for SPIRE development at IFSI. This could be ensured by procurement of an additional DPU or use of HIFI DPU with a test engineer.

Regarding project management

SP-IF-N-5 Additional manpower needs to be added to SPIRE ICU software development. The usage of additional resources needs to be co-ordinated with HIFI and PACS needs at IFSI. To gain maximal use of additional manpower, areas capable of largely independent development (e.g., software testing, user manual preparation) need to be identified.

SP-IF-N-6 Planning for maintenance and operations needs to be done and included in the SUM. In this regard:

- Tools and documents needed for software maintenance need to be specified; those applicable to a particular release need to be documented;
- Tools, e.g. simulations or flight spare (FM cold part and QM warm part) need to be available;
- The contents of the software release note for major releases need to be defined;
- Expanded use of the configuration tool, e.g. to insert data in headers and provide the indication of the configuration status for files in a release, needs to be studied;
- Specific knowledge of developers needs to be available to the project;
- A repeatable, semi-automated software build procedure needs to be implemented and described.

Note that it is a concern that RAL will not have a duplicate maintenance environment to the one to be retained at IFSI.

SP-IF-N-7 Planning for FM needs to be completed. In this regard:

- The FM baseline and any additional EEPROM patches need to be finalised;
- The format for monitoring tables input from RAL needs to be defined and agreed;
- A document delivery schedule needs to be produced.

SP-IF-N-8 Major releases need to be consistent with the instrument schedule. In this regard:

- Regression tests need to be performed, but the number of repetitions needs to be planned with reference to the schedule. Access to the RAL system may be needed to ensure consistency;
- AVM test requirements need to be ascertained;
- An intermediate delivery tailored for AVM test requirements (i.e., simplified autonomy software) needs to be considered.

SP-IF-N-9 Simulations used for the CDMS, instrument subsystems and the DRCU need to be validated.

SP-IF-N-10 Software modifications need to be done correctly. In this regard:

- Experience of 'bug' history at IFSI needs to be captured;
- The final software version used in a model needs to be fully tested.

SP-IF-N-11 Reporting needs to be improved, i.e.,

- Software integration problems need to be communicated to the instrument team;
- IFSI testing and test progress needs to be made more visible to the instrument team;
- Suspected DPU HW problems need to be visible to all affected parties;
- Special means need to be considered to allow IFSI to participate in Working Group discussions, if engineers are unable to attend meetings;
- MRB or equivalent needs to be held for SPR closeout.

SP-IF-N-12 Duplicate effort among IFSI HIFI, PACS and SPIRE engineers regarding the ASI (i.e., Carlo Gavazzi) interface needs to be reduced.

Regarding documentation (see also §4.4.2)

SP-IF-N-13 Elements of concurrent design need to be captured in the documentation, i.e.,

- Intertask dependencies and forced sequencing need to be documented;
- Use of flags, sequencing and other mechanisms need to be documented to ensure no potential deadlock is present in code or introduced in code during maintenance;
- Data protection mechanisms need to be described.

SP-IF-N-14 Constraints and agreements not part of requirements or interface documents need to be signalled in the code or in the SUM.

Regarding software engineering

SP-IF-N-15 The source code needs to be coherent, i.e.,

- In the absence of detailed design documentation, a code navigation aid needs to be available;
- Coding standards (for example, to ensure a consistent level of meaningful commenting) need to be defined and applied;
- External variables need to be signalled as external where they are used;
- FDIR must match current FDIR philosophy, i.e., housekeeping information needs to be reported consistently.

SP-IF-N-16 Verification coverage and repeatability of development tests need to be assessed.

4.4.2 Missing or insufficient documentation

The following documents are missing and need to be produced with contents in line with the checklists of /5/.

- SUM
- SVVP

The following documents are considered insufficient and need to be produced with contents in line with the checklists of /5/.

- SSD (= combined SRD/ADD)
- SCIDL

These four documents should comprise a minimal document set for the SPIRE ICU software.

4.5 Risk Analysis

4.5.1 Overview

A risk analysis according to the methods of /10/ was performed. Severity and likelihood categories were identified and applied to the identified risks as shown in Figure 4.5-1.

4.5.2 Results

The results of risk management are summarised in Figure 4.5-1, which identifies the risks, associated needs, likelihood, severity and proposed management strategy.

ID	Title	Need	Like	Sev	Status
1	*Regarding hardware*				
1.1	System integration problems due to inadequate lower-level testing of software	N1 / N3	L	CR	mitigate
1.2	Acceptance tests unrealistic or unsuitable	N2	M	M	mitigate
1.3	IFSI gets insufficient use of HW for development	N4	M	CR	mitigate
2	*Regarding project management*				
2.1	IFSI unable to meet schedule with current manpower	N5	VH	S	mitigate
2.2	IFSI does not produce plan for operations and maintenance	N6	M	O	track
2.3	FM definition is too late	N7	VH	S	mitigate
2.4	Overall schedule is not tenable	N8	VH	S	mitigate
2.5	Systematic error not detected due to simulation error	N9	L	M	mitigate
2.6	Software bugs introduced during modification	N10	L	M	mitigate
2.7	Problem resolution is unnecessarily retarded	N11	H	S	mitigate
2.8	Effort duplicated between HIFI/PACS/SPIRE project	N12	VH	O	track
3	*Regarding documentation*				
3.1	SSD incomplete for final delivery	N13 / 4.4.2	H	CR	mitigate
3.2	SCIDL incomplete for final delivery	4.4.2	L	O	track
3.3	SUM incomplete for final delivery	N14 / 4.4.2	M	O	track
3.4	SVVP incomplete for final delivery	4.4.2	H	M	mitigate
4	*Regarding software engineering*				
4.1	Design features not apparent in code	N15	H	O	track
4.2	Code coverage incomplete or tests not repeatable	N16	H	M	mitigate

Key

Need = Nx/4.4.2, where 4.4.2 refers to section 4.4.2 and Nx refers to SP-IF-N-x in section 4.4.1.

Like = Likelihood

Very High:	Probability estimated to be higher than 95%
High:	Probability estimated to be between 65% and 95%
Medium:	Probability estimated to be between 25% and 65%
Low:	Probability estimated to be between 5% and 25%
Very Low:	Probability estimated to be under 5%

Sev = Severity

CAtastrophic	Leads to abandonment of project by developer
CRitical:	Leads to a substandard or nonworking product
Major:	Leads to a product with degraded functionality
Significant:	Leads to schedule delay of more than one month
Operational:	Leads to extra work during operations/maintenance
Negligible:	Minimal or no impact

Status = suggested strategy for risk management

mitigate:	Risk is to be addressed by acting to satisfy the corresponding need(s)
track:	Risk is to be monitored at Instrument level; action if suggested by re-evaluation of probability or severity
accept:	Risk is to be accepted

Figure 4.5-1 Summary of Risk Analysis Results

Regarding the risks identified for mitigation, the following priorities and dependencies are noted:

The risks assigned a critical severity should be given priority over those with lower severity.

Among the risks assigned a critical severity, risk 3.1 should take precedence over risk 1.3 and risk 1.3 over 1.1, i.e., risks are ranked in order of probability.

The risks assigned a major severity should be given priority over those with lower severity.

Among the risks assigned a major severity, risk 3.4 and 4.2 should take precedence over risk 1.2 and risk 1.2 over risks 2.5 and 2.6, i.e., risks are ranked in order of probability.

In order to get maximum use out of additional resources, Risk 2.4 should take priority over Risk 2.3 and Risk 2.3 should take priority over Risk 2.1. All three of these should take priority over Risk 2.7, since the probability of Risk 2.7 is lower.

This can be presented graphically as follows

Highest Priority

↑

| 3.1

| |

| 1.3

| |

| 1.1

| 3.4 4.2

| |

| 1.2

| |

| 2.5 2.6

| 2.4 2.3

| |

| 2.1

| |

| 2.7

↓

Lowest Priority

APPENDIX A: AUDIT QUESTIONNAIRE RESPONSES

A.1 Audit Questionnaires

A.1.1 Questions for the Instrument Developer

Question	Instruction/Clarification	Response
1. How are instrument-level and System-level requirements divided between hardware and software components?		High-level software requirements are extracted from the Instrument Requirements Document (SPIRE-RAL-PRJ-000034) by the Instrument Scientist and included in the URD (SPIRE-IFS-PRJ-000444) after discussion with IFSI.
2. How are software autonomy requirements (e.g., FDIR), evident perhaps only at instrument level or following hardware design review (or hardware delivery!), captured and communicated to software developers?		Supplementary requirements documents giving more details on autonomy requirements (SPIRE-RAL-PRJ-001855), and peak-up mode requirements (SPIRE-RAL-PRJ-001969) have been written after review of the Hardware-Software interaction document.
3. How are interfaces managed, i.e., (a) interface with CDMS, (b) interface between different instrument subsystems, and (c) interfaces between independent teams working on the same instrument subsystem?		Interfaces are detailed Interface Control Documents: the DPU Interface Document (SPIRE-IFS-PRJ-000650) for the DPU electrical interfaces; the Packet Structure ICD for the 1553 data interface; and the DRCU/DPU Interface Control Document (SPIRE-SAP-PRJ-001324) for the data interface to the DRCU. These document are under configuration control.
4. What configuration management requirements apply to software that is common to different instruments or different instrument subsystems?		The SPIRE on-board software is treated as if it were independent of the other instruments. We have no visibility of common components. We expect IFSI to handle configuration control of all software modules. It is their responsibility to ensure that any common modules use the latest version.

Question	Instruction/Clarification	Response
5. How is the content of various software versions specified and how is verification of different versions controlled? Has delivered software met stated needs and requirements?		Three main versions of the software to be delivered were considered in the development plan. IFSI are responsible for unit and system-level testing of all software running in the DPU. An acceptance test has been carried out on the initial delivery of the first version (the only version delivered so far). SPRs raised at that time have been addressed and the current release of the first version is adequate for the initial testing of the instrument.
6. How do you track software progress?		Reporting by IFSI of progress versus requirements and closure of SPR/SCRs.
7. How do you plan to address software operations and maintenance phases?		IFSI will remain responsible for all software maintenance during the operations phase. They will maintain the code generation tools, trained staff and resources for generating and testing new OBS software as required. New software delivered by IFSI, will be tested by the ICC Operations Centre at RAL (using simulators and possibly the Flight Spare instrument model) before being delivered to the Ground Segment for upload to the satellite.
8. Have software interface, performance and operational requirements been based on actual instrument (or System) needs? Has delivered software met these requirements and needs?		These requirements (given in the URD) were derived from the instrument requirements. The software is tested against these requirements.

Question	Instruction/Clarification	Response
9. If a problem is identified with a supplied hardware or software item, how are other parties working on H-P notified of the problem?		As far as the SPIRE project is concerned we are formally not aware of any common components and so there is no formal mechanism for notification. If a problem would affect the spacecraft an NCR is created. It is expected that within IFSI there is a mechanism for problems identified on one system to be passed on to those working on other systems. Informally the instrument Project Managers are in regular contact and pass on information about problems identified with their systems.
10. Who is responsible for software acceptance test definition and implementation, and which requirements are verified during acceptance tests?		IFSI is responsible for the definition of the Acceptance test (subject to RAL approval). The tests are carried out at RAL using more representative test equipment. The Acceptance Test should test the user requirements (including autonomy requirements, peak-up mode requirements and any additional functionality included through SCRs).
11. Are the interfaces with other participating institutes (or companies) and with ESA adequate for obtaining necessary inputs and for resolving problems?		All contributing institutes have limited resources and for some this has meant that information is difficult to obtain. In particular it has been difficult to specify the DPU-DRCU data interface. The interface with IFSI subcontractors has been difficult to penetrate. Within the IFSI-CGS contract it has not always been possible to get information from their suppliers. With limited resources IFSI have been unable to attend some relevant working group which would have enabled a more timely resolution of problems.

A.1.2 Questions on Software Development

Question	Instruction/Clarification	Response
1. What is the mechanism for requirements change? Do you have an agreed requirements baseline?	Requirements baseline refers to software requirements defined at the level of the user of the software and detailed descriptions of interfaces, protocols the software must provide or use. Mechanisms could involve re-issue of URD or ICD, change request, etc.	The baseline requirements are presently collected in three documents; i) OBS URD (SPIRE-IFS-PRJ000444), ii) Autonomy Functions Requirements Document (SPIRE-RAL-PRJ001855) and iii) Peak-up Mode Requirements Document (SPIRE-RAL-PRJ001969). Each new requirement or modification of old requirement is negotiated and, if agreed, would produce an update in the relevant document.
2. What levels of testing (e.g., software unit-level, instrument or subsystem-level with simulator, instrument or subsystem-level with real hardware) are defined for your software, i.e., does the practice correspond to the description of PA Plan, section 4.3? What is verified at each level?	Functional and performance requirements must be verified at some level of testing.	Planned documented testing is currently outlined in the SVVP; from the document you can see that only system level (and partially integration level) testing is being documented. Unit level testing has not been documented because the code is still evolving and keeping documented testing up with the development would have required more resources in the software team.
3. How and at what stages in software development do you make sizing and timing estimates and measurements? What are the estimates for the complete software?	Timing refers to both CPU usage and time to complete specific tasks, e.g., memory dump of maximum size. There are various ways of estimating sizing and timing, i.e., extrapolation from prototype or previous project, line or operation counts, etc. Ideally budgets and estimates should be defined according to a decomposition of the software into smaller units.	The CPU load is reported in one of the parameters of the HK packets, only with the purposes of monitoring the system activities. No timing requirements have been foreseen up to now for the completion of some specific task (e.g. the memory dump of maximum size) and therefore no time measurements for the execution of the commands have been performed, except for simple monitoring purposes during tests in case of problems.
4. When in your development process is your software placed under configuration control and how are changes handled after that, e.g., are all changes traced to SPR?	Configuration control mechanisms should be appropriate to the project and allow the status of delivered and planned releases to be determined.	The software is currently placed under configuration control, even if no official delivery have been done yet. Whenever a version of the OBS is delivered it is extracted from the repository and tagged opportunely with a revision number. There is an SPR/SCR server active at ESA that we are currently using; upon each OBS delivery a note is provided with the list of fixed SPRs/SCRs.

Question	Instruction/Clarification	Response
5. What is the makeup of your development team, i.e., size, experience, roles?		The HIFI software development team is composed by two part-time persons: Sergio Molinari, PhD in Astronomy, previous experiences in the design and development of software for data analysis. He is working on the SPIRE software since 2001. His duties involve the interface to the project team, OBS design, development and testing supervision, document preparation. John Liù, student in Software Engineering. Experienced in C and Assembler. He is working on SPIRE OBS since 2002 and is responsible for code development and testing. Upon request, the following persons help the team for special topics: Internal interpreter implementation and HW/SW integration: R. Cerulli , degree in Physics, great experience in development of software for space applications.
6. Where software is developed by a team, how is it ensured that information is disseminated and that the resulting product is (sufficiently) complete and uniform?	This could involve meetings and reviews, internal document list, etc.	The work of the two team persons do not overlap.
7. What experience has your organisation had with similar projects? What experience have the individual team members had with similar projects?		IFSI has a great expertise in space experiments. IFSI groups participated to the development of several payload instruments for astronomical ESA (and non-ESA) space missions. The infrared astronomy group in particular, which has in charge the ICU provision, participated to ESA ISO by providing the onboard DPU and the OBS for the ISO LWS instrument. As stated in point 5, the team member with a previous experience in similar software problems is Dr. R. Cerulli, who participated to the HW/SW provision for several experiments on board ESA missions.

Question	Instruction/Clarification	Response
8. What development environment and tools for software development (e.g., design, compilation, configuration management and test) do you use? What capabilities do the tools have, e.g., • Do development tools allow the capability to perform assignment of memory to fixed addresses at linker symbol and module level, so that data and function addresses can be frozen? • Do development and test tools allow the capability to extract timing and CPU usage information? • Do test tools allow the capability to perform test coverage analysis, automated retest and differencing? • Do test tools provide a means for comparing expected against actual test outputs? • Do configuration management tools allow the extraction of full configuration history for each file? • Do configuration management tools enforce separation of released code from code under development? • Do configuration management tools support the grouping of files into modules to allow easier extraction for building and module testing?	Tools should be appropriate to the project and can be in-house items.	Design: AxiomSys CASE tool, provided by Structured Technology Group, Inc., based on the Structured Analysis method. Compilation: VIRTUOSO + ADI 21020 Ctools. Editor: Context. Configuration management: Tortoise CVS. Test: Subsystem simulators provided by Stockholm observatory; RAL provided CDMS simulator; RAL provided packet display tool; in house developed very simple tools for the TM data analysis.

Question	Instruction/Clarification	Response
9. What is the estimated level of completeness of software development tasks, e.g. (a) requirements analysis, (b) architectural design, (c) detailed design, (d) coding, (e) verification, (f) operations and maintenance planning? How are the results of these tasks documented?	The actual task breakdown depends on software development method.	Requirement analysis: completed to the 100%. Architectural design: completed to the 90%. Detailed Design: N/A, according to the OBS PA plan. Coding: completed to the 90%. HW/SW integration testing: completed to the 100%. Verification: completed to the 70%. Operations and maintenance planning: no planning at all, except for a general agreement about the OBS maintenance responsibility and implementation scenario. Documentation of the tasks results: completed to the ?%.
10. Can you identify planned dates for completing the above development tasks, the inputs needed and the associated risk factors?	Risk factors tend to be specific to the development and usually relate to open areas and design drivers.	See the SPIRE OBS Continuous Management Plan. Deliveries are on schedule for the moment.
11. What are the principal open areas that remain in your software development? When and how do you plan to resolve them?		The main open areas are the partial implementation and testing of the autonomy functions and the peak-up mode. Their implementation is planned with the scheduled deliveries.
12. What metrics (e.g., lines of code, estimated vs. actual effort) do you use to monitor and control development? Do you feel confident based on trends and available resources that the schedule can be met? If not, what would help?	Metrics do not have to involve tools or be quantitative.	No metrics is used.
13. Are the interfaces with other participating institutes (or companies) and with ESA adequate for obtaining necessary inputs and for resolving problems?		Yes.

Question	Instruction/Clarification	Response
14. What are the mechanisms for (technical) interface control? How effective are they? Is the level of collaboration/consultation with software engineers in defining interfaces (e.g., event and housekeeping packet definitions) sufficient?		The technical interface control is the result of the collaboration of RAL (Ken King) with IFSI on one side and with the other subsystem developers on the other side. Every need in terms of changes in the packets definitions and contents has been discussed and agreed all together.
15. What are the mechanisms for software review and progress monitoring practiced by ESA and by the instrument team? How effective are they?		Since the CDR no other official review have been done with ESA and the Consortium. Since the development is on schedule, the instrument team is not currently monitoring the progress in the software development.

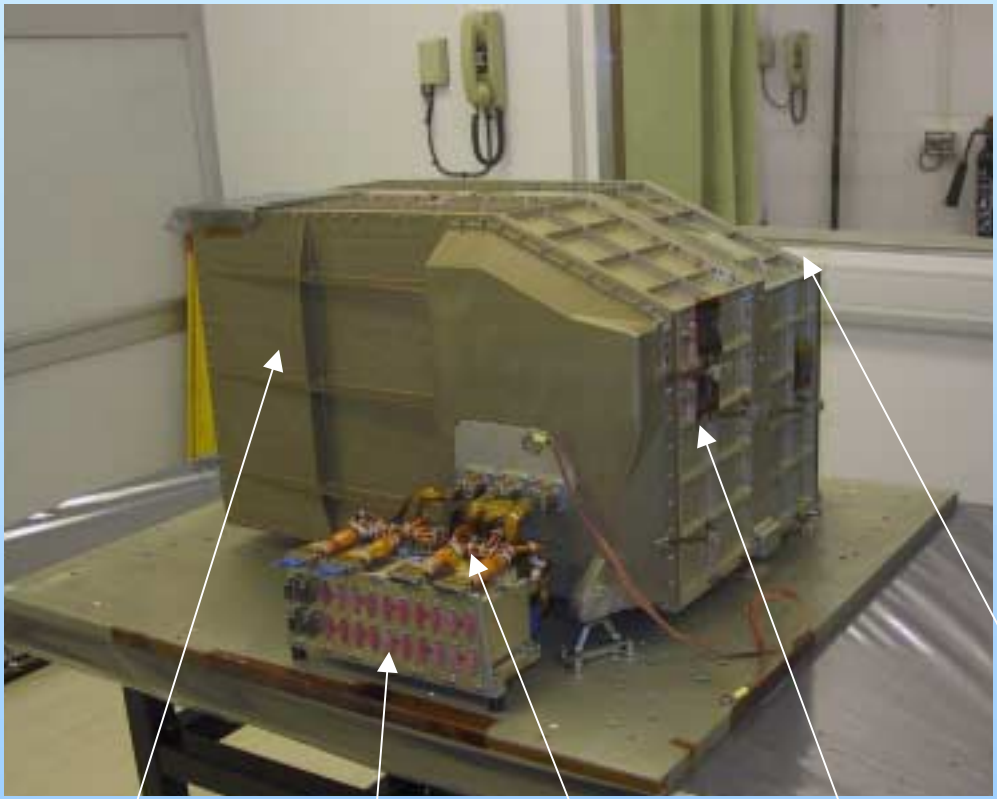
A.1.3 Questions on Software Documentation

Question	Instruction/Clarification	Response
1. In OBS-UR-TM5, should new data overwrite old data if the buffer overflows?	This should be notified to ESA.	No new memory blocks are allocated if the overflow condition is reached; so no overwrite, but data are lost anyway.
2. In OBS-UR-TM16, where does the requirement for HK TM with frequency up to 5 Hz come from?		I guess it was from an agreement with the project.
3. Is it possible to make OBS-UR-AF8 more specific, i.e., what does 'fast response times' mean?		This should be quantified by the project.
4. For which TC will the start and end of process etc. be reported (ref. OBS-UR-AF11)?		This is determined by the value of the "Ack bits" contained in the standard header of the TC that originates the action.
5. Is OBS-UR-AF12 covered by specific requirements?		Not yet, I believe.
6. Where are the maximum data rates referenced in OBS-SUR-FU11 defined?		The origin of this requirement is erroneously, I believe, assigned at IFSI. The estimated rates were communicated to us by the project.
7. Can a higher priority interrupt be handled during the service of another, lower-priority interrupt?		Yes.
8. In the SEG_DMDA design what happens if allocation fails to find a large enough contiguous block?		Blocks are allocated within pools (separate for HK, TC, Science and Events packets) whose dimensions is properly sized (e.g., to meet OBS-UR-TM5) fixed at initialization. The unavailability of blocks within pools means we entered overflow condition.
9. The software design appears to be very coupled. Was it considered to design objects around the various tables and pools of data to limit coupling to particular modules?		Please clarify.

Question	Instruction/Clarification	Response
10. How long is the maximum outage before data may be lost, i.e., how long do SPU etc. need to buffer data in case of DPU outage?	This should be notified to other subsystem.	As far as we know, the subsystems are not able to buffer data.
11. Have the pre-defined procedures in detecting of anomalies been defined (refer to TBD in OBS-CUR-AF2 of the URD)?	Experience suggests that insufficient or incorrect FDIR definition introduces considerable uncertainty into software design and sizing/timing estimates and can cause significant rework.	These are documented in the Autonomy Functions Requirement Document.
12. The requirements of the URD seem to be traced directly to design, with no logical model in between. What information is used to provide requirements detail not provided in the URD?		This step has not been documented.
13. Where are default values of housekeeping parameters defined?		They are contained in on-board tables available at startup and updatable via TC.
14. Where is information for software maintenance defined, i.e., building of executable, use of detailed memory map for patching data, constant or program areas?		This is only partially contained in the OBS user manual.

APPENDIX B: PRESENTATIONS

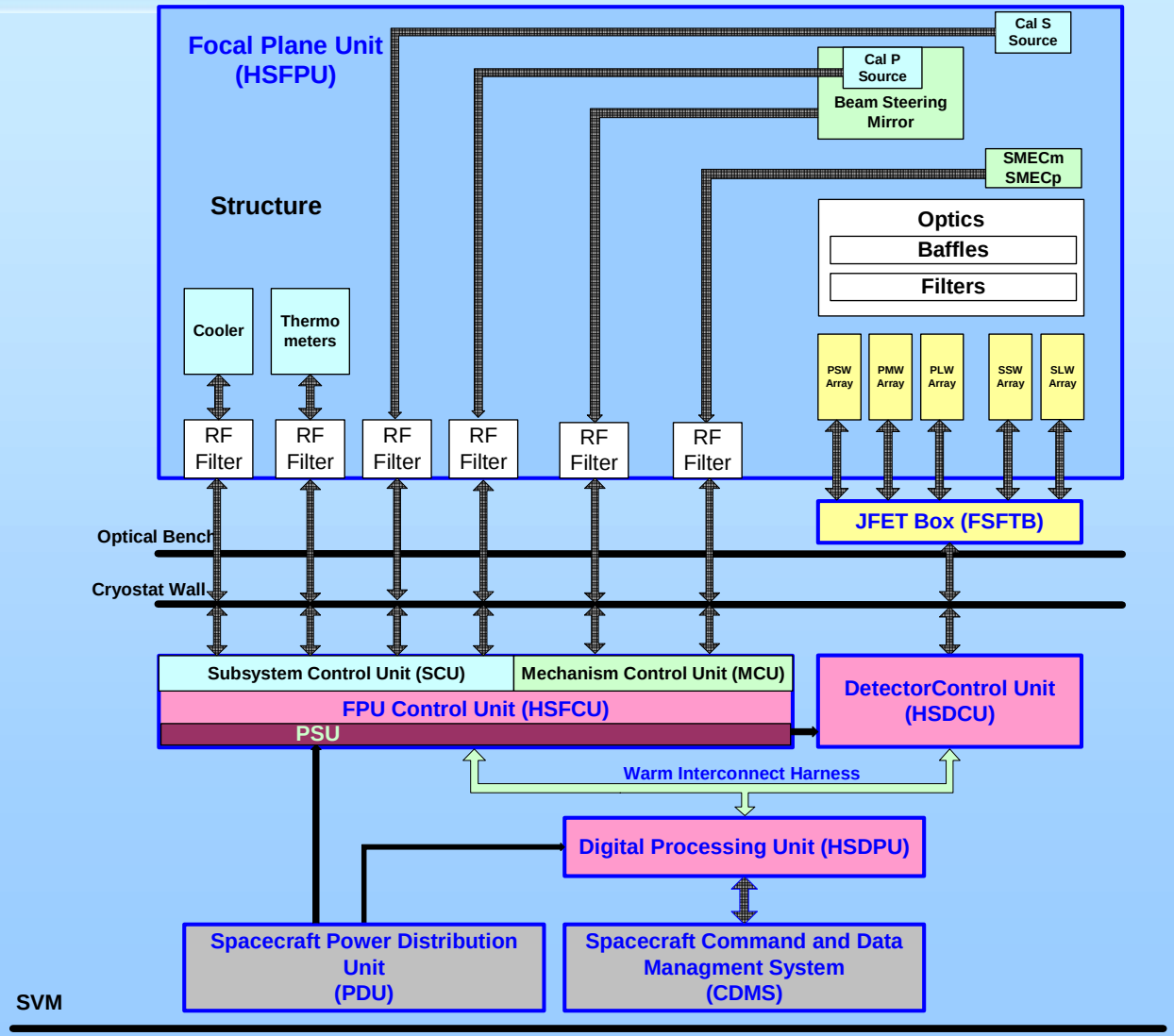
SPIRE Instrument



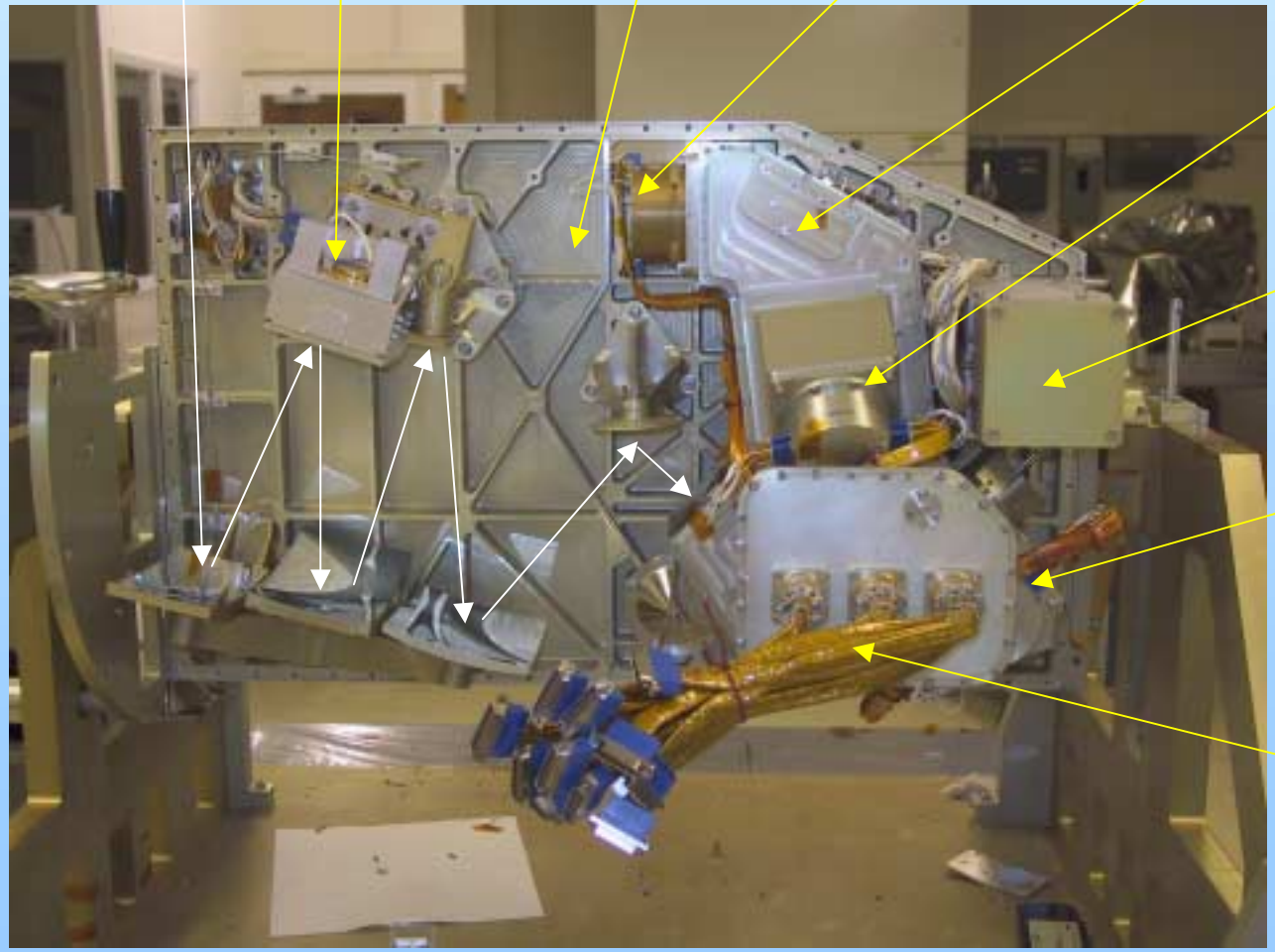
Biographical Details

- **S**pectrographic and **P**hotometric **I**maging **RE**ceiver
- Photometer simultaneously images three bands: 250, 360 and 520µm
- Spectrometer images in two bands: 200-325 µm and 315-670 µm
- Instrument Field of View: 4'x8'
- Optimised to map large swathes of the sky in three bands (colours)
- Follow up sources at higher photometric or spectrographic sensitivities

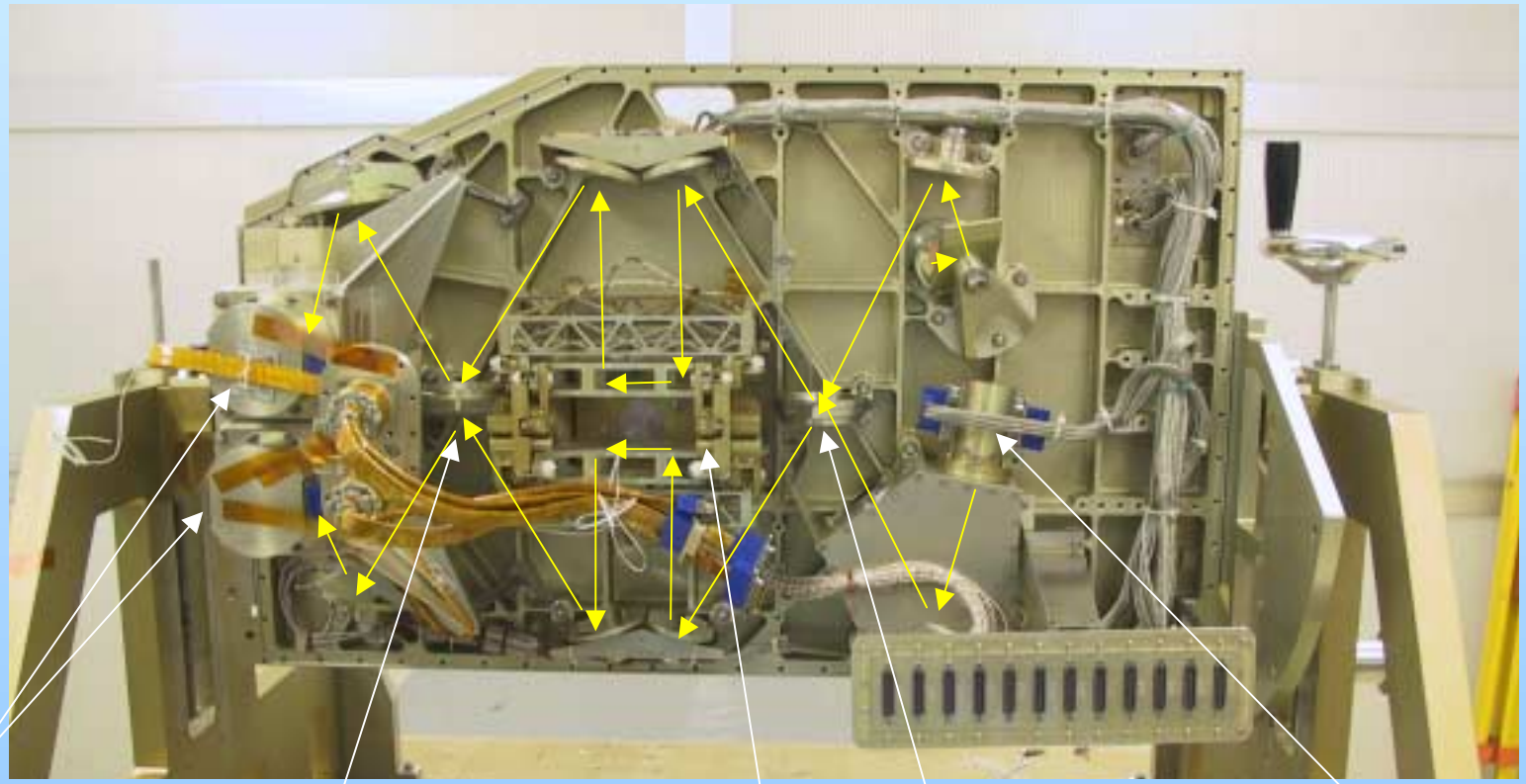
SPIRE Block Diagram



SPIRE Photometer

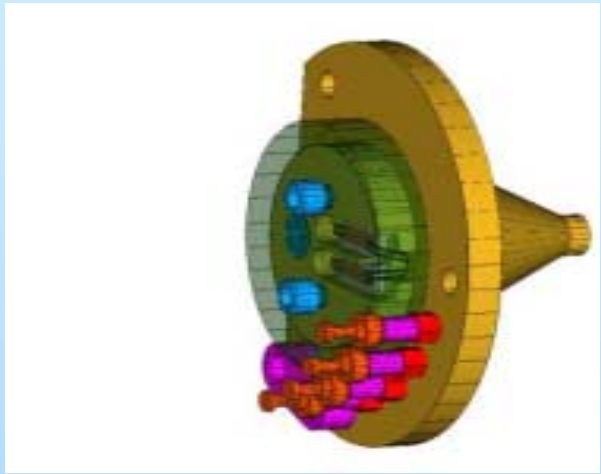


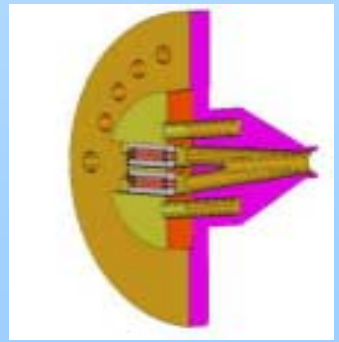
SPIRE Spectrometer

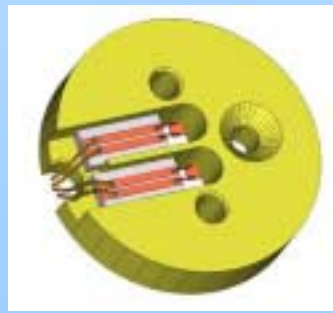


Photometer Calibrator (PCAL)

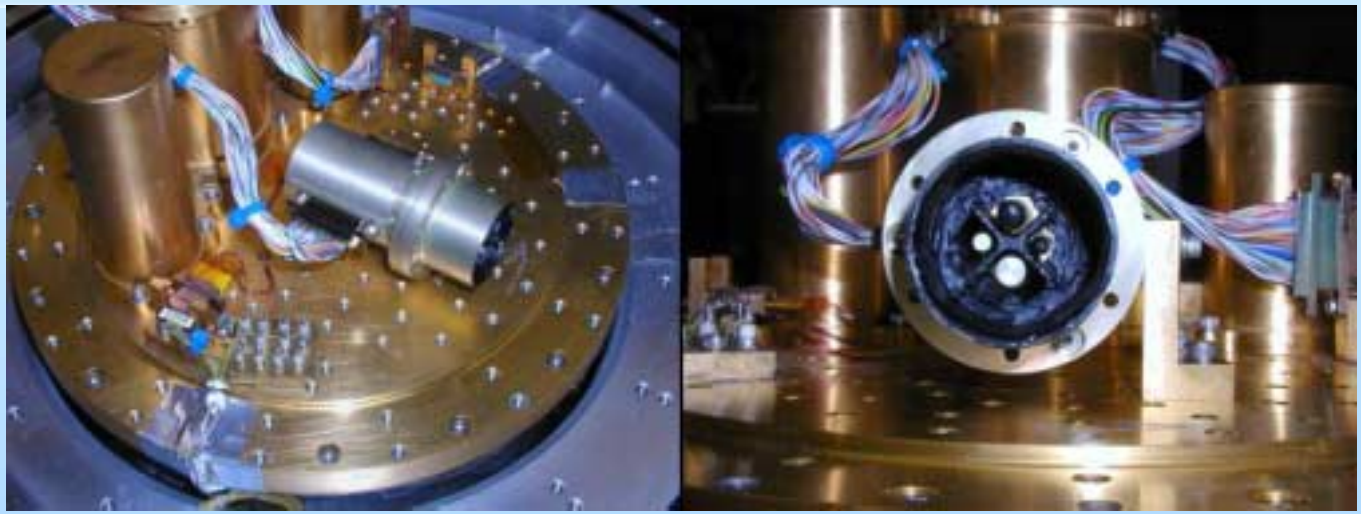
- **Use**
  - To monitor changes in detector responsivity by illuminating all detectors with a known flux and measuring the response
- **Inputs**
  - Heater Current
- **Outputs**
  - Measured Heater Current
- **Operations**
  - PCAL_Flash
    - Switch current with square wave input (0 – 4Hz) – modulated signal







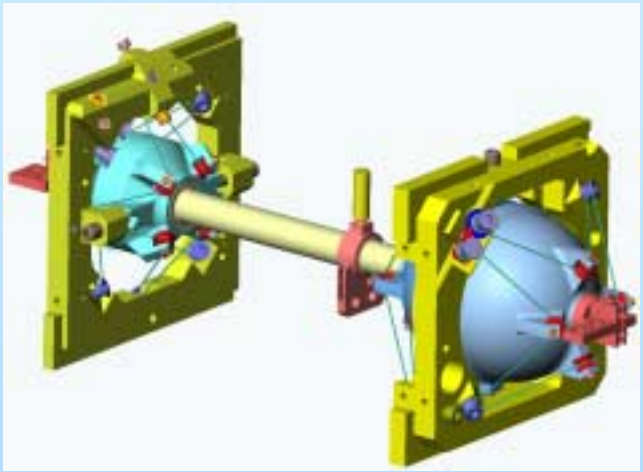
Spectrometer Calibrator (SCAL)

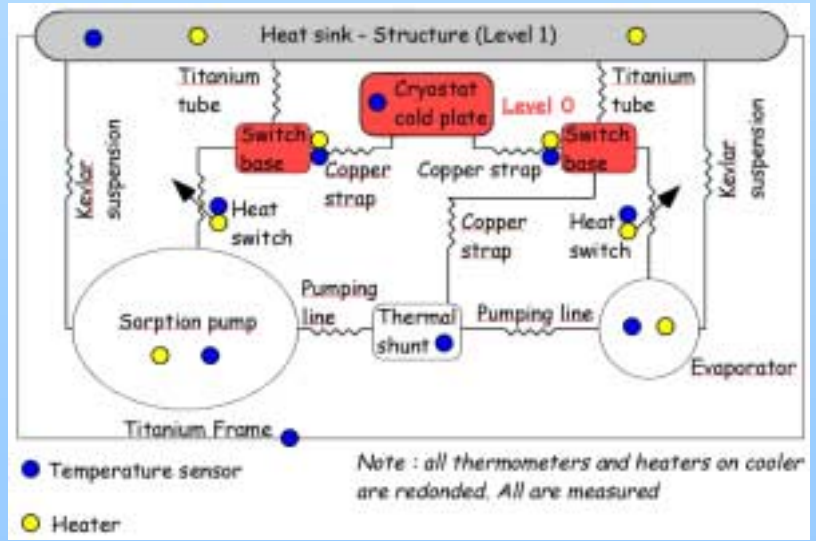


- **Use**
  - To offset spectral signal from telescope
- **Inputs**
  - Heater Current(s)
- **Outputs**
  - Measured Heater Current
  - Operating Temperature
- **Operations**
  - SCAL_Control
    - PID controller to achieve stable temperature

Sorption Cooler

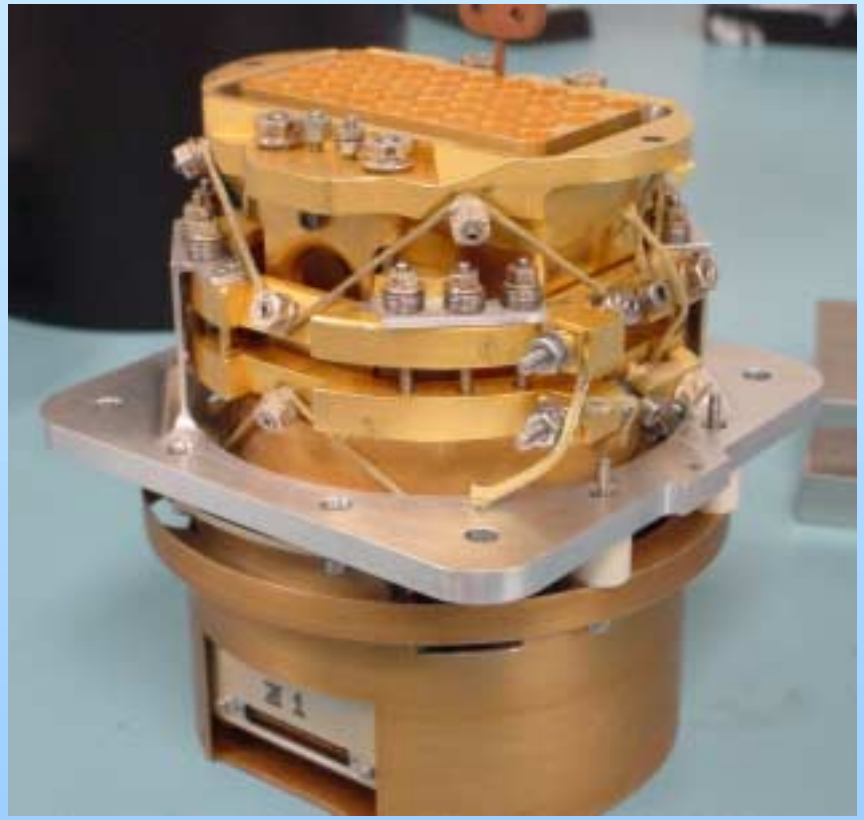
- **Use**
  - Maintain the detectors at their operating temperature (300mK)
- **Inputs**
  - Heater Currents
- **Outputs**
  - Measured Heater Current
  - Temperatures
- **Operations**
  - Cooler_Recycle
    - Heat pump, cool evaporator to drive He_3 into evaporator





Detectors

- Use
  - To detect flux from source
- Inputs
  - Configuration parameters (bias, JFET voltage etc)
- Outputs
  - Measured 300mK temperature
  - TM: Photometer or Spectrometer signals
  - TM: Photometer or Spectrometer offset values
- Operations
  - Det_Thermal_Control
    - PID control to stabilise detector temperature
    - Time constant ~ 10s of seconds



Beam Steering Mirror (BSM)

- **Use**
  - Modulate the signal (chop) in order to determine source flux on top of large background
  - To move input beam across FOV (jiggle) in order to sample beam adequately
- **Inputs**
  - Chop and Jiggle position
- **Outputs**
  - Measured drive currents, and position
  - TM: Position and currents
- **Operations**
  - Chop
    - Move between two positions in chop direction
    - Up to 2 Hz
    - Take detector and BSM data at each position
  - Chop and Jiggle (TBC)
    - Move to up to 256 positions in chop and jiggle directions and perform chop at each position

Spectrometer Mechanism (SMEC)

- **Use**
  - Spectroscopy
- **Inputs**
  - Configuration parameters
  - Move to position
  - Scan n times
- **Outputs**
  - Configuration parameters
  - TM: time & stage position
- **Operations**

  none

Photometer Observing Modes

Observing Modes	POF1	POF2	POF3	POF4	POF5	POF6	POF7	POF8
DPU		**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**
Essential Hsk packets	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Normal Hsk packets	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TC Acceptance	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Event packets	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Science packets	Phot_Full	Phot_Full	Phot_Full	Phot_Full	Phot_Full	Phot_Full	Phot_Full	Phot_Full
VM	Chop	Chop_and_Jiggle	Chop_and_Jiggle	Chop_and_Jiggle		Chop	Chop_and_Jiggle	Flash
VM1	Det Temp Ctrl	Det_Temp_Ctrl	Det_Temp_Ctrl	Det_Temp_Ctrl	Det_Temp_Ctrl	Det_Temp_Ctrl	Det_Temp_Ctrl	Det_Temp_Ctrl
VM2								
VM3		Autonomy	Autonomy	Autonomy	Autonomy	Autonomy	Autonomy	Autonomy
MODE	0x0310	0x0320	0x0330	0x0340	0x0350	0x0360	0x0370	0x0380
DRCU		**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**
SCU								
Temp Channels powered	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SubK Channel powered	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PCAL source powered								Yes
SCAL sources powered								
Cooler TC Heater powered	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Cooler SP Heater powered								
Cooler EV HS powered								
Cooler SP HS powered	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DCU								
Photometer JFET Htrs powered								
Photometer BIAS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Photometer JFETS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Photometer LIAs	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Spectrometer JFET Htrs powered								
Spectrometer BIAS								
Spectrometer JFETS								
Spectrometer LIAs								
MCU								
DSP	Operational	Operational	Operational	Operational	Operational	Operational	Operational	Operational
BSM	Chop	Chop & Jiggle	Chop & Jiggle	Chop & Jiggle	Hold	Chop	Chop & Jiggle	Hold
SMEC								

Spectrometer Observing Modes

Observing Modes	SOF1	SOF2	SOF3	SOF4	Parallel	Peak-up
DPU	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**
Essential Hsk packets	Yes	Yes	Yes	Yes	Yes	Yes
Normal Hsk packets	Yes	Yes	Yes	Yes	Yes	Yes
TC Acceptance	Yes	Yes	Yes	Yes	Yes	Yes
Event packets	Yes	Yes	Yes	Yes	Yes	Yes
Science packets	Spec_Full	Spec_Full	Spec_Full	Spec_Full	Phot_Parallel	Phot_Full
VM			Step_and_Look	Step_and_Look		Peak-up
VM1					Det_Temp_Ctrl	Det_Temp_Ctrl
VM2	SCAL Ctrl	SCAL Ctrl	SCAL Ctrl	SCAL Ctrl		
VM3	Autonomy	Autonomy	Autonomy	Autonomy	Autonomy	Autonomy
MODE	0x0410	0x0420	0x0430	0x0440	0x0500	0x0321
DRCU	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**
SCU						
Temp Channels powered	Yes	Yes	Yes	Yes	Yes	Yes
SubK Channel powered	Yes	Yes	Yes	Yes	Yes	Yes
PCAL source powered						
SCAL sources powered	Yes	Yes	Yes	Yes		
Cooler TC Heater powered					Yes	Yes
Cooler SP Heater powered						
Cooler EV HS powered						
Cooler SP HS powered	Yes	Yes	Yes	Yes	Yes	Yes
DCU						
Photometer JFET Htrs powered						
Photometer BIAS					Yes	Yes
Photometer JFETS					Yes	Yes
Photometer LIAs					Yes	Yes
Spectrometer JFET Htrs powered						
Spectrometer BIAS	Yes	Yes	Yes	Yes		
Spectrometer JFETS	Yes	Yes	Yes	Yes		
Spectrometer LIAs	Yes	Yes	Yes	Yes		
MCU						
DSP	Operational	Operational	Operational	Operational	Operational	Operational
BSM	Hold	Hold	Chop	Chop	Hold	Chop
SMEC	Scan	Scan	Step	Step		

SPIRE Configurations

	OFF	INIT	DPU ON	DRCU ON	REDY	PHOT STRT	PHOT STBY	PHOT SEREN	SPEC STRT	SPEC STBY	SPEC SEREN	CREC	SAFE
DPU		**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**
Essential Hsk packets			0.5Hz	0.5Hz	0.5Hz	0.5Hz	0.5Hz	0.5Hz	0.5Hz	0.5Hz	0.5Hz	0.5Hz	0.5Hz
Normal Hsk packets			1.0Hz	1.0Hz	1.0Hz	1.0Hz	0.25Hz	1.0Hz	1.0Hz	0.25Hz	1.0Hz	1.0Hz	1.0Hz
TC Acceptance			Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Event packets		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Science packets						Phot_Offsets		Phot_Serendipity	Spec_Offsets		Spec_Serendipity		
VM												Cooler_Recycle	
VM1						Det_Temp_Ctrl	Det_Temp_Ctrl	Det_Temp_Ctrl	Det_Temp_Ctrl	Det_Temp_Ctrl	Det_Temp_Ctrl		
VM2													
VM3			Autonomy	Autonomy	Autonomy	Autonomy	Autonomy	Autonomy	Autonomy	Autonomy	Autonomy	Autonomy	Autonomy
MODE			0x0000	0x0100	0x0200	0x0203	0x0300	0x0390	0x0204	0x0400	0x0490	0x0600	0x0900
DRCU				**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	
SCU													
Temp Channels powered					Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
SubK Channel powered					Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
PCAL source powered													
SCAL sources powered													
Cooler TC Heater powered						Yes	Yes	Yes					
Cooler SP Heater powered												Yes	
Cooler EV HS powered												Yes	
Cooler SP HS powered					Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
DCU													
Photometer JFET Htrs powered						Yes							
Photometer BIAS						Yes	Yes	Yes					
Photometer JFETS						Yes	Yes	Yes					
Photometer LIAs						Yes	Yes	Yes					
TC BIAS													
TC JFETS						Yes	Yes	Yes					
TC LIAs													
Spectrometer JFET Htrs powered									Yes				
Spectrometer BIAS									Yes	Yes	Yes		
Spectrometer JFETS									Yes	Yes	Yes		
Spectrometer LIAs									Yes	Yes	Yes		
MCU													
DSP					Boot	Operational	Operational	Operational	Operational	Operational	Operational	Operational	
BSM						Switch On	Hold	Hold	Switch On	Hold	Hold		
SMEC									Init	Standby	Standby		

Additional Operations

- **Peak-up Mode**
  - Chop at a series of points in the form of a cross
  - Calculate signal difference at each point
  - Identify maximum signal
  - Generate TM packet to request telescope to move to put maximum signal at centre of array
- **Autonomy actions**
  - Determined by FDIR
    - Issue event packet
    - Stop VMs
    - Inhibit commands

OBS Constraints

- **Telecommands**
  - From mission timeline – accuracy ~ 1sec
  - TC rate < 2 per second
  - Total number of telecommands limited by on-board storage
  - OBS must expand TC to instrument commands
    - Operational modes
    - Housekeeping data collection
- **Telemetry**
  - Limited number of TM packets (24) per second – independent of packet size
  - OBS packs SPIRE data into TM packets to maximise throughput

Command Lists and their Execution

- **Command Lists**
  - **Provide updateable functionality**
    - **Changes required to meet different environments**
    - **Upgrades to operations in the light of better understanding**
    - **Safer than OBS code changes**
      - Restricted set of allowed commands
      - Restricted access to memory and other OBS resources
  - **Allow SPIRE team to upgrade OBS functionality without requiring OBS recompilation and upload**
    - **Saves time for implementation**
    - **Saves uplink time (whole OBS has to be uploaded)**
- **VMs**
  - **Interpret Command Lists**
  - **VM**
    - **High priority interrupt driven – time resolution of few microsecs**
    - **Used for data taking operations**
  - **VM1,2,3**
    - **Uses virtuoso for timing – few millisec resolution**
    - **Uses high priority queue for sending commands – can be delayed by VM and other non-interrupt driven VMs**
    - **Used to implement thermal control and autonomy actions**

OBS Deliveries

- OBS consists of Boot Software and Application Software
- Software deliveries;
  - ILT Version
    - Boot Software
    - Functionality for testing at instrument level
  - AVM version
    - Includes autonomy functions
    - Allows testing of operations command lists
    - Includes peak-up operation
  - Launch Version
    - Includes default contents for tables and operations command lists
- Testing
  - CGS tests Boot software – IFSI Accepts?
  - IFSI carries out unit/system testing of code using CDMS/DRCU simulators
  - Acceptance test of delivered code performed at RAL, with CDMS Simulator and DRCU (simulator for ILT version)

OBS Maintenance

- All problems are reported by issue of an SPR (uses SPR system maintained by HSC)
- IFSI analyse SPR and suggest solution
  - May need to raise an NCR if H/W related
- After agreement IFSI implement the solution
- IFSI issue upgrade to OBS periodically containing all implemented SPRs
- RAL close the SPRs when their functionality has been proven
- Changes to Specification are handled through SCRs using same system

Open Areas

- **Code visibility**
  - No code reviews – design information only available from Architectural design
  - No coding standard – individual modules written in differing styles with varying amounts of information
- **Configuration Control**
  - Code is apparently controlled using CVS, but no indication in modules of their version number
  - No indication which modules and which versions of these are included in any delivered version of OBS
- **Documentation**
  - Needs to be strengthened by final delivery to ensure maintenance is possible after launch - by anyone, we cannot ensure continuation of availability of staff.
- **Testing**
  - It is not clear what unit-level and system-level tests are carried out
  - Lack of test reports

SPIRE On-Board Software Status

IFSI-CNR

SPIRE OBS Team

- Sergio Molinari
  - Astronomer
  - Some software experience and common sense
  - Duties
    - Overall responsability of the OBS
    - Interface to project team at RAL
    - OBS design & architecture
    - Documentation
    - Testing Supervision
- Riccardo Cerulli-Irelli
  - Engineer
  - Noble Father
  - Duties
    - Interface design
    - Virtual Machine idea, design, implementation & testing
- Scigè John Liù (aka Nino)
  - Student in Software Engineering
  - Thinks and communicates in HEX
  - Duties
    - Coding
    - Testing

SPIRE OBS Documentation

- Requirements
  - URD v1.2 - May 15th 2003
  - Autonomy Functions (still not frozen)
  - Peak-up mode (still not frozen)
- Design
  - ADD v1.0 - May 18th 2003
    - Architectural design
    - Requirement Traceability Matrix
    - Needs minor revision
- Testing
  - SVVP v1.3 - June 18th 2003
    - Testing of high-level OBS functions (e.g. task intercommunication)
    - Acceptance tests
    - Needs minor revision
- OBS User Manual (currently v0.9)
  - up-issued at every OBS major release

SPIRE OBS Status (1/2)

- We are working according to an OBS developement plan
- First delivery of OBS version 1.0 to RAL on April 2003
  - Successive deliveries 1.1a to 1.1d
    - Bug fixes
    - New requirements
- OBS Version 1.2 delivered November 25th 2003
  - Successive deliveries 1.2a to 1.2i (as we speak)
    - Bug fixes
    - New Requirements
  - Cumulative fixes for more than 20 SPRs/SCRs
  - Under testing at IFSI (with DRCU simulator) and at RAL (with real DCU-MCU-SCU!).

SPIRE OBS Status (2/2)

- OBS Version 1.3 will be delivered at X+4 months (X is the date when requirements are frozen)
  - New components
    - Autonomy functions
    - Peak-up mode
  - Requirements are 80% frozen. Implementation has started accordingly
    - Only partial testing done so far.
- Version 1.4 delivery planned for Nov 2004, but probably shifting a bit following 1.3
  - Implementing a standard set of Virtual Machine programs for different observing modes
  - Need specs

SPIRE OBS Problem Areas

- Adherence to PA Plan
- Documentation
  - Anything more than updates to current issues will be a problem
- Testing
  - Not documented at low-level
  - Full OBS verification in a variety of scenarios (including full verification of VM)
  - Full FDIR verification