TITLE:

Planck LFI DPC – Configuration Items List

DOC. TYPE: **LIST**

PROJECT REF.: **PL-LFI-OAT-LI-003** **PAGE:** **I of IV, 66**

ISSUE/REV.: **1.1** **DATE:** **June 2008**

Issued by	**M. FRAILIS** **F. GASPARO** **D. MAINO** **M. REINECKE** **R. ROHLFS** **LFI DPC Development Team**	**Date:** **Signatures:**	**JUNE 16, 2008**
Agreed by	**F. PASIAN** **LFI DPC PA Manager** **A. ZACCHEI** **LFI DPC Manager**	**Date:** **Signature:**	**JUNE 16, 2008**
Approved by	**R.C. BUTLER** **LFI Program Manager**	**Date:** **Signature:**	**JUNE 16, 2008**
Approved by	**N. MANDOLESI** **LFI Principal Investigator**	**Date:** **Signature:**	**JUNE 16, 2008**

DISTRIBUTION LIST

Recipient	Company / Institute	E-mail address	Sent
T. PASSVOGEL	ESA/ESTEC/SCI-PT	Tpassvog@estec.esa.nl	Yes
P. ESTARIA	ESA/ESTEC/SCI-PT	Pestaria@estec.esa.nl	Yes
J. TAUBER	ESA/ESTEC/SA	Jtauber@astro.estec.esa.nl	Yes
N. MANDOLESI	IASF/CNR – BOLOGNA	Mandolesi@iasfbo.inaf.it	Yes
R. C. BUTLER	IASF/CNR – BOLOGNA	butler@iasfbo.inaf.it	Yes
M. BERSANELLI	UNIMI – MILANO	marco@ifctr.mi.cnr.it	Yes
C. LAWRENCE	JPL – PASADENA	Lawrence.a.wade@jpl.nasa.gov	Yes
F. PASIAN	OAT – TRIESTE	Pasian@oats.inaf.it	Yes
K. BENNETT	ESTEC/SSD – NOORDWIJK	Kbennet@astro.estec.esa.nl	Yes
L. DANESE	SISSA – TRIESTE	Danese@sissa.it	Yes
G. DE ZOTTI	OAPd – PADOVA	Dezotti@oapd.inaf.it	Yes
K. ENQVIST	UN. of HELSINKI	Kari.enqvist@helsinki.fi	Yes
E. KOLLBERG	CUT – GOTHENBURG	kollberg@ep.chalmers.se	Yes
E. MARTINEZ-GONZALEZ	UN. de CANTABRIA – SANTANDER	Martinez@ifca.unican.es	Yes
H.-U. NORGAARD-NIELSEN	DSR – COPENHAGEN	hunn@dsri.dk	Yes
R. REBOLO	IAC – LA LAGUNA TENERIFE	Rrl@ll.iac.es	Yes
J. TUOVINEN	MIL – HELSINKI	jussi.tuovinen@vtt.fi	Yes
S. WHITE	MPA – GARCHING	Swhite@MPA-Garching.MPG.DE	Yes
J.-L. PUGET	HFI – IAS	puget@ias.u-psud.fr	Yes
J. CHARRA	HFI – IAS	charra@ias.u-psud.fr	Yes
G. EFSTATHIOU	HFI – CAM	gpe@ast.cam.ac.uk	Yes
F. BOUCHET	HFI – IAS	bouchet@ias.u-psud.fr	Yes
A. ZACCHEI	INAF-OATs	Zacchei@oats.inaf.it	Yes

CHANGE RECORD

Issue	Date	Sheet	Description of Change	Release
0.1	20 Sept 2004		First initial draft (table of contents)	
0.2	24 Sept 2004	All	Included software CIDL for Pipeline DM	SGS-DR
1.0	15 Dec 2006	All	Revision, include OM	SGS-IR
1.1	16 June 2008	All	General Revision; included configured hardware	SGS-RR

TABLE OF CONTENTS

1 SCOPE

This document provides information on “as designed” and “as built” configuration items relative to S/W and H/W in the preparation of the LFI DPC. For S/W, it contains information at the level of DPC Level and module: each software configuration item (source, object, executable) is listed together with build instructions, used tools (with their version), and input/output parameters lists, applicable requirement documents, list of all raised and incorporated software SPRs/RIDs. For H/W, it contains information on each hardware configuration item.

This document contains the following information:

- Lists of all hardware configuration items
- Lists of all software configuration items (source, object, executable) with the version of all their constituents (components)
- List of all software build instructions and used tools (with their version)
- List of all test files reports
- Modules descriptions and input/output parameters lists (preferably, as XML files compliant with IDIS Process Coordinator specifications [AD-4])
- Applicable requirement documents
- List of all incorporated/raised software SPRs/RIDs.

1.1 LIMITS OF APPLICABILITY

This document describes all DPC hardware and software items put under configuration management including:

- Mission Simulator (“Level S”),
- Pipeline, including “Level 1” (telemetry processing), “Level 2” (TOI processing), “Level 3” (scientific processing), “Level 4” (data delivery)

It does not cover IDIS products, which will be handled separately by the IDIS developers. It may include acquired COTS up to the level needed to list other configured items.

Procedures can be simplified with respect to those defined in [AD-6].

2 APPLICABLE/REFERENCE DOCUMENTS

2.1 APPLICABLE DOCUMENTS

[AD-01] – ECSS Q80, Space Project Management – Software Product Assurance
[AD-02] – ECSS M40, Space Project Management – Configuration Management
[AD-03] – LFI Science Operation Implementation Plan, PL-LFI-OAT-PL-001
[AD-04] – LFI Configuration and Data Management Plan, PL-LFI-PST-PL-001
[AD-05] – LFI Management Plan, PL-LFI-PST-PL-005
[AD-06] – LFI DPC Software Configuration Management Plan, PL-LFI-OAT-PL-002
[AD-07] – LFI DPC Pipeline Level 2 and Level 3 URD, PL-LFI-OAT-UR-006
[AD-08] – LFI DPC TQL URD, PL-LFI-OAT-UR-003
[AD-09] – LFI DPC TMH URD, PL-LFI-OAT-UR-004
[AD-10] – Planck LFI – LIFE: URD, PL-LFI-OAT-UR-009

2.2 REFERENCE DOCUMENTS

[RD-01] – Planck LFI DPC Software Policy, PL-LFI-OAT-SD-001
[RD-02] – Access Policy to Planck/LFI DPC CVS Software Repositories, PL-LFI-OAT-SD-002
[RD-03] – IDIS Software Component URD, PL-COM-SSD-UR-003
[RD-04] – IDIS Process Coordinator URD, PL-COM-MPA-UR-001
[RD-05] – Compilation and usage of Planck Simulation Modules
[RD-06] – Compilation and Usage of the LFI-DPC Level 2 Modules, PL-LFI-OAT-MN-xxx
[RD-07] – Planck LFI DPC Hardware Design, PL-LFI-OAT-SP-005

2.3 ACRONYMS LIST

CI	Configuration Item
CIDL	Configuration Item Data List
CMP	Configuration Management Plan
CMT	Configuration Management Tool
COTS	Commercial Off The Shelf Software
CSCI	Computer Software Configuration Item
DPC	Data Processing Centre
DMC	Data Management Component
DMS	Documentation Management SYSTEm
IDIS	Integrated Data and Information System
ESTEC	European Space Technology and Research Centre
IO	Instrument Operations
MOC	Mission Operations Centre
NCR	Non Conformance Report

OAT	Osservatorio Astronomico di Trieste
QLA	Quick Look Analysis
RID	Review Item Discrepancy
RTA	Real Time Assessment
SCCB	Software configuration control board
SCM	Software Configuration Management
SCR	Software Change Request
SPDP	Software Project Development Plan
SPA	Software Product Assurance
SPR	Software Problem Report
SSD	Space Science Department (ESTEC)
TM	Telemetry
TOI	Time Ordered Information

3 LEVEL S - CONFIGURATION LIST

The complete source code is located in the directory planck/LevelS/ in ESTEC's CVS repository.

Extensive documentation of the parameters accepted and data sets read/written by the modules is contained in the XML descriptions located in planck/LevelS/xml.

The file “testdata.tar.gz” can be obtained at:
http://planck.mpa-garching.mpg.de/SimData/test/testdata.tar.gz

Detailed documentation can be found in the document “Compilation and usage of Planck Simulation Modules” [RD-5].

For every module (executable) a full description is given.

Module Name:	cmb	Version:	Source:	
			Executable:	cmb
Module Description:	CMBfast code for generation of power spectra			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name:	syn_alm_cxx	Version:	Source:	
			Executable:	syn_alm_cxx
Module Description:	converts a power spectrum to a realisation of spherical harmonic coefficients			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name:	alm2map_cxx	Version:	Source:	
			Executable:	alm2map_cxx
Module Description:	converts spherical harmonic coefficients to HEALPix map			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			

Test Data Files:	contained in testdata.tar.gz
I/O Parameters:	
Applicable URDs :	
SPRs/RIDs:	

Module Name:	anafast_cxx	Version:	Source:	
			Executable:	anafast_cxx
Module Description:	converts a HEALPIX map into spherical harmonic coefficients			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name:	almmixer	Version:	Source:	
			Executable:	almmixer
Module Description:	combines spherical harmonics of CMB and foregrounds			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name:	skymixer	Version:	Source:	
			Executable:	skymixer
Module Description:	combines HEALPix maps of CMB and foregrounds			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name:	gaussbeampol	Version:	Source:	

			Executable:	gaussbeam pol
Module Description:	generates spherical harmonics for an elliptic Gaussian polarised beam			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name:	beam2alm	Version:	Source:	
			Executable:	beam2alm
Module Description:	converts a GRASP8 beam description to spherical harmonic coefficients			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name:	alm2grid	Version:	Source:	
			Executable:	alm2grid
Module Description:	converts spherical harmonics to values on a geographic grid			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name:	totalconvolve_cxx	Version:	Source:	
			Executable:	totalconvolve_cxx
Module Description:	performs total convolution on spherical harmonics of sky and beam			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				

Applicable URDs :	
SPRs/RIDs:	

Module Name:	simmission	Version:	Source:	
			Executable:	simmission
Module Description:	generates satellite pointing information from a scanning strategy			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name:	multimod	Version:	Source:	
			Executable:	multimod
Module Description:	creates time-ordered data and co-added maps of the scanned sky signal, dipole, 1/f-noise, point-source signal and sorption-cooler noise. The effect of the detector electronics is simulated			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name:	map2tga	Version:	Source:	
			Executable:	map2tga
Module Description:	visualisation tool for HEALPix maps			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:	contained in testdata.tar.gz			
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name:	ls2lfitoi	Version:	Source:	Ls2lfitoi.cc
			Executable:	ls2lfitoi

Module Description:	Create LFI toi (with Sky and Load signals) starting from output from LevelS. These should include sky signal, white noises, differenced and un-differenced 1/f noises as well as thermal variations on both telescope and 4K load.
Build Instructions:	Follow instructions in planck/LevelS/README.compilation
Test Data Files:	
I/O Parameters:	
Applicable URDs :	
SPRs/RIDs:	

Module Name:	quantum	Version:	Source:	
			Executable:	quantum
Module Description:	Implement the effect of quantization for the LFI data			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:				
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

Module Name	Exchange_30(44,70,hk)	Version:	Source:	
			Executable:	exchange_30(44,70,hk)
Module Description:	Produce data at 30, 44, and 70 GHz (TOI) and Housekeeping in the exchange data format reading input data from Database			
Build Instructions:	Follow instructions in planck/LevelS/README.compilation			
Test Data Files:				
I/O Parameters:				
Applicable URDs :				
SPRs/RIDs:				

4 LEVEL 1 - CONFIGURATION LIST

For every module a full description is given. The section is divided by main Level 1 components.

4.1 EXTERNAL COMPONENTS

Module Name:	ROOT	Version: 5.18.00	Source:	
			Executable:	
Module Description:	C++ library			
Build Instructions:	export ROOTSYS=/opt/root configure make export PATH=$ROOTSYS/bin:$PATH export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH			
Test Data Files:				
I/O Parameters:				
Applicable URDs :	Planck LFI- Telemetry Quick-Look URD			
SPRs/RIDs:				

Module Name:	toodi	Version: 2.5.0	Source:	
			Executable:	
Module Description:	Java library, with C and fortran wrappers, to read from/write to the DMC database			
Build Instructions:	make make newKodoOracle (for details see the user manual)			
Test Data Files:				
I/O Parameters:				
Applicable URDs :	Planck LFI Telemetry Handling System URD, Planck IDIS Data Management Component URD			
SPRs/RIDs:				

Module Name:	cfitsio	Version:	Source:	

		2.490.2	Executable:	
Module Description:	Library to read and write FITS files. Provided within the Level 1 software distribution.			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:				
I/O Parameters:				
Applicable URDs :	Planck LFI- Telemetry Quick-Look URD, Telemetry Handling System URD			
SPRs/RIDs:				

Module Name:	readline	Version: 4.2.1	Source: Executable:	
Module Description:	library to edit input strings. Provided within the Level 1 software distribution.			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:				
I/O Parameters:				
Applicable URDs :	Planck LFI Telemetry Handling System URD			
SPRs/RIDs:				

Module Name:	fftw	Version: 2.1.3	Source: Executable:	
Module Description:	Library to calculate the fast fourier transorm. Provided within the Level 1 software distribution.			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:				
I/O Parameters:				

Applicable URDs :	Planck LFI- Telemetry Quick-Look URD
SPRs/RIDs:	

Module Name:	tinyxml	Version: 2.4.3	Source:	
			Executable:	
Module Description:	Library to read and write XML files. The library contains customizations performed by the ISDC team and is provided within the Level 1 software distribution			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:				
I/O Parameters:				
Applicable URDs :	Planck LFI- Telemetry Quick-Look: URD, Telemetry Handling System - URD			
SPRs/RIDs:				

4.2 LIBRARIES

Module Name:	makefiles	Version: 2.9.3	Source:	
			Executable:	
Module Description:	a utility to build the Level 1 software components			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:				
I/O Parameters:				
Applicable URDs :	Planck LFI- Telemetry Quick-Look URD, Telemetry Handling System URD			
SPRs/RIDs:				

Module Name:	pil	Version: 1.9.11	Source:	
			Executable:	
Module Description:	a library to access program parameters			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:				
I/O Parameters:				
Applicable URDs :	Planck LFI Telemetry Handling System URD			
SPRs/RIDs:				

Module Name:	ril	Version: 4.0.1	Source:	
			Executable:	
Module Description:	library to write log messages and alerts			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:				
I/O Parameters:				
Applicable URDs :	Planck LFI Telemetry Handling System URD			
SPRs/RIDs:				

Module Name:	templates	Version: 2.5	Source:	
			Executable:	
Module Description:	a list of cfitsio template files, describing the FITS extensions			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:				
I/O Parameters:				

Applicable URDs :	Planck LFI- Telemetry Quick-Look URD, Telemetry Handling System URD
SPRs/RIDs:	

Module Name:	dal	Version: 4.0.1	Source:	
			Executable:	
Module Description:	library for creation and manipulation of hierarchical data sets spanning multiple files and abstracting the file format used.			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:				
I/O Parameters:				
Applicable URDs :	Planck LFI Telemetry Handling System - URD			
SPRs/RIDs:				

Module Name:	gui_library	Version: 2.9	Source:	
			Executable:	
Module Description:	an extension to ROOT for the display programs (TQL)			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:				
I/O Parameters:				
Applicable URDs :	Planck LFI- Telemetry Quick-Look: URD			
SPRs/RIDs:				

Module Name:	RTSILib	Version: 1.0	Source:	
			Executable:	
Module Description:	library encapsulating the implementation of the client side of the RTSI interface.			

Build Instructions:	export OMNIORB4_INCLUDE=<custom OMNIORB include directories> export OMNIORB4_LIBS=<custom directories for the OMNIORB libraries> make idl make
Test Data Files:	l1_archive_SOVT1a_RT.tgz, HFA_SOVT1a_RT.tgz
I/O Parameters:	
Applicable URDs :	HPMCS Science Ground Segment Real-time Telemetry ICD
SPRs/RIDs:	

4.3 TMH PROGRAMS

Module Name:	tm_request	Version:1.9	Source: Executable:	
Module Description:	The program sends requests for catalog, telemetry, Telecommand History and time correlation data to the MOC DDS.			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a.tgz			
I/O Parameters:				
Applicable URDs :	Planck LFI Telemetry Handling System URD, HPMCS Data Disposition System ICD			
SPRs/RIDs:				

Module Name:	tm_receive	Version:1.0	Source: Executable:	
Module Description:	This client application connects to the MOC RTSI server and receives the near real-time telemetry. It dispatches the telemetry data according the Application Process ID and writes out the dispatched data every DataSetDuration minutes. It feeds the TQL display(s) with the data sequentially			

Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)
Test Data Files:	l1_archive_SOVT1a_RT.tgz
I/O Parameters:	
Applicable URDs :	PLANCK LFI Telemetry Handling System URD, Planck LFI Telemetry Quick-Look URD, HPMCS Science Ground Segment Real-time Telemetry ICD
SPRs/RIDs:	

Module Name:	tm_merge	Version:2.0	Source:	
			Executable:	
Module Description:	The program reads one DDS input file containing the requested TM data and converts it into a FITS file with the raw TM packets, merging it with already existing FITS file covering the same time range if needed.			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a.tgz			
I/O Parameters:				
Applicable URDs :	Planck LFI Telemetry Handling System URD, HPMCS Data Disposition System ICD			
SPRs/RIDs:				

Module Name:	tmu	Version:1.7	Source:	
			Executable:	
Module Description:	The program reads one FITS file of TM data of the same application process ID and produces the final product which is the Time Ordered Information(TOI) data in FITS format.			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a.tgz			

I/O Parameters:	
Applicable URDs :	Telemetry Handling System - URD
SPRs/RIDs:	

Module Name:	hk_conversion	Version:1.1	Source:	
			Executable:	
Module Description:	converts hk data into physical units			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a.tgz			
I/O Parameters:				
Applicable URDs :	Telemetry Handling System - URD			
SPRs/RIDs:				

Module Name:	sc_calibration	Version:1.0	Source:	
			Executable:	
Module Description:	converts scientific data into physical units			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a.tgz			
I/O Parameters:				
Applicable URDs :	Telemetry Handling System - URD			
SPRs/RIDs:				

Module Name:	aux_extraction	Version:2.2	Source:	
			Executable:	
Module Description:	converts the attitude history file to FITS file in the raw data repository			

Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)
Test Data Files:	l1_archive_SOVT1a.tgz
I/O Parameters:	
Applicable URDs :	Telemetry Handling System - URD
SPRs/RIDs:	

Module Name:	limit_check	Version: 1.0	Source:	
			Executable:	
Module Description:	checks all HK data for predefined limits			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a.tgz			
I/O Parameters:				
Applicable URDs :	Telemetry Handling System - URD			
SPRs/RIDs:				

Module Name:	scet_check	Version: 1.0	Source:	
			Executable:	
Module Description:	tests the correlation of SCET time and the OBT time of each TM packet compared to the time correlation records.			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a.tgz			
I/O Parameters:				
Applicable URDs :	Planck LFI Telemetry Handling System URD			
SPRs/RIDs:				

Module Name:	fits2dmc	Version: 2.1.1	Source:	
			Executable:	
Module Description:	copy data from FITS files into DMC tables			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a.tgz			
I/O Parameters:				
Applicable URDs :	Planck LFI Telemetry Handling System URD, Planck IDIS Data Management Component URD			
SPRs/RIDs:				

Module Name:	am_daemon	Version: 1.1.1	Source:	
			Executable:	
Module Description:	writes the alert information into the DPC centralize alert log files			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a.tgz			
I/O Parameters:				
Applicable URDs :	Planck LFI Telemetry Handling System URD			
SPRs/RIDs:				

Module Name:	tch2ascii	Version: 1.0	Source:	
			Executable:	
Module Description:	converts the DDS command history data into an ASCII format			

Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)
Test Data Files:	l1_archive_SOVT1a.tgz
I/O Parameters:	
Applicable URDs :	HPMCS Data Disposition System ICD
SPRs/RIDs:	

Module Name:	time_corr_merge	Version: 1.5	Source:	
			Executable:	
Module Description:	reads the time correlation coefficient "TM packet" received from MOC via the DDS system and stores the important data into the time correlation FITS file			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a.tgz			
I/O Parameters:				
Applicable URDs :	Planck LFI Telemetry Handling System URD, HPMCS Data Disposition System ICD			
SPRs/RIDs:				

4.4 TQL PROGRAMS

Module Name:	tql_desktop	Version: 1.3	Source:	
			Executable:	
Module Description:	Start-up display of the TQL programs: it provides functionalities to open an existing TM archive, receive real-time telemetry from the tm_receive program, start all the other TQL displays			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			

Test Data Files:	
I/O Parameters:	
Applicable URDs :	Planck LFI- Telemetry Quick-Look URD
SPRs/RIDs:	

Module Name:	mode_display	Version: 1.1.2	Source:	
			Executable:	
Module Description:	shows the available science data for all detectors of all horns for a defined time range			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a_RT.tgz			
I/O Parameters:				
Applicable URDs :	Planck LFI- Telemetry Quick-Look URD			
SPRs/RIDs:				

Module Name:	monodim_display	Version: 1.2.2	Source:	
			Executable:	
Module Description:	displays scientific data			
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)			
Test Data Files:	l1_archive_SOVT1a.tgz, l1_archive_SOVT1a_RT.tgz			
I/O Parameters:				
Applicable URDs :	Planck LFI- Telemetry Quick-Look URD			
SPRs/RIDs:				

Module Name:	hk_display	Version: 1.3.1	Source:	
			Executable:	

Module Description:	displays HK data: timelines, histograms and correlation plots
Build Instructions:	tools/makefiles/ac_stuff/configure make clean make global_install (for details see the user manual)
Test Data Files:	l1_archive_SOVT1a.tgz, l1_archive_SOVT1a_RT.tgz
I/O Parameters:	
Applicable URDs :	Planck LFI- Telemetry Quick-Look URD
SPRs/RIDs:	

4.5 RTSICLIENT (FOR SCOS 3.1 EGSE)

Module Name:	addon_rtsiclient_rep	Version: 1.1	Source:	
			Executable:	
Module Description:	Client application developed within the SCOS framework. It connects to the MOC RTSI server and receives the near real-time telemetry, forwarding it to the SCOS PDS.			
Build Instructions:	To be built within the SCOS development environment: source .pmakews build-rep (see the Readme.txt instructions for details)			
Test Data Files:	HFA_SOVT1a_RT.tgz			
I/O Parameters:				
Applicable URDs :	HPMCS Science Ground Segment Real-time Telemetry ICD			
SPRs/RIDs:				

4.6 LIFE, RANA AND RACHEL

Module Name:	Rachel	Version:2.4	Source:	S/W packet
			Executable:	rachel
Module Description:	Rachel (Radiometer Chains Evaluator) is a open source software designed to perform high-speed on-line analysis and data storage for data coming from either network or local sockets. It has been implement for store data from the RCA test of the LFI instrument.			

Build Instructions:	There is a configuration file that set up the Makefile for the complete package. It requires the Qt libraries (>3.0). It is written in C++.
Test Data Files:	
I/O Parameters:	• **Output:** o TABLE: DAT file: file containing the data from the RCA (after DAE BB) with alternate sky-load signal for each detector. o TABLE: ENV file: file with all environment condition (thermal sensors). In genereal H/K. o TABLE: CFG file: configuration file of both the cryo-chamber used for the test and for the instrument itself (bias current and voltages applied to each device – amplifier, diode, phase-switch) o TABLE: LOG file: log file for tracing the whole path of the test
Applicable URDs :	
SPRs/RIDs:	

Module Name:	Rana	Version:1.0	Source:	IDL modules
			Executable:	Rana
Module Description:	RaNA (Radiometer aNAlyser, which, by the way means "frog" in italian), is a code written in IDL 6.0 which will be used as a post-processing tool of the radiometric data produced during the QM and FM RCA calibration and test campaigns of the Planck-LFI instrument.			
Build Instructions:	None. It requires IDL 6.0. Otherwise it is possible to have a single executable file with all RaNA functionality			
Test Data Files:				
I/O Parameters:	• **Input** o TABLE: DAT file: file containing the data from the RCA (after DAE BB) with alternate sky-load signal for each detector. o TABLE: ENV file: file with all environment condition (thermal sensors). In genereal H/K. o TABLE: CFG file: configuration file of both the cryo-chamber used for the test and for the instrument itself (bias current and voltages applied to each device – amplifier, diode, phase-switch) o TABLE: LOG file: log file for tracing the whole path of the test • **Output** o Reports with speficic radiometer parameters values after test.			
Applicable URDs :	[AD-10]			
SPRs/RIDs:				

Module Name:	LIFE	Version:2.0	Source:	IDL modules and CC modules

			Executable:	Life
Module Description:	LIfE (Lfi Integrated perFormance Evaluator), is a code written in c++ and IDL 6.0 which will be used as a post-processing tool of the radiometric data produced during the QM and FM RAA calibration and test campaigns of the Planck-LFI instrument. It will be used too during operations to create the required daily Report and weekly report.			
Build Instructions:	None. It requires IDL 6.0. Otherwise it is possible to have a single executable file with all LIfE functionality			
Test Data Files:				
I/O Parameters:	• **Input** o Housekeeping timelines o Science timelines o Science Calibration file (to transform from ADU to Volts) o AHF file o Data quality report XML structure o Weekly quality report xml structure • **Output** o DQR o WHR o Reports with speficic radiometer parameters values after test.			
Applicable URDs :	[AD-10]			
SPRs/RIDs:				

5 LEVELS 2+3 - CONFIGURATION LIST

For every module (executable) a full description is given. There are general rules for compilation. These are specified in the config/config_`mymachine` file. Here it an example for the DPC machine with Linux OS and gfortran/gcc/g++ compilers. In this example the variable `$(COMLIB)` and `$(COMINC)` refers to the include and libraries directories in the Level S package.

```
MOD = mod
FC= gfortran -std=gnu -fno-second-underscore
FL= gfortran -std=gnu -fno-second-underscore
F_MODPATH= -I
F_DEFINE= -D
F_PORTABILITY_FLAGS = -DPLANCK_GFORTRAN
F_COMMONFLAGS= -W -Wall -Wno-uninitialized -O2 -Wfatal-errors
FCFLAGS = -c $(F_COMMONFLAGS) -I$(INCDIR) -ffree-line-length-none -fimplicit-non
e -I$(COMINC)
FLFLAGS = -L. -L$(LIBDIR) -s -L$(COMLIB)
CC= gcc
CCFLAGS_NO_C= -W -Wall -Wno-long-long -I$(INCDIR) -fno-strict-aliasing -O2 -g0 -
s -ffast-math -fomit-frame-pointer -Wfatal-errors -Wno-unknown-pragmas -Wshadow
-Wmissing-prototypes --std=gnu89 -pedantic -I$(COMINC)
CCFLAGS= $(CCFLAGS_NO_C) -c -L$(COMLIB)
CXX= g++
CXXL= g++
CXXWARNFLAGS= -Wall -Wextra -Wstrict-aliasing=2 -Wundef -Wshadow -Wwrite-strings
 -Wredundant-decls -Woverloaded-virtual -Wcast-qual -Wcast-align -Wpointer-arith
 -Wold-style-cast -Wno-unknown-pragmas
CXXCFLAGS_NO_C= $(CXXWARNFLAGS) -ansi -I$(INCDIR) -O2 -g0 -s -ffast-math -fomit-
frame-pointer -Wfatal-errors -I$(COMINC)
CXXCFLAGS= $(CXXCFLAGS_NO_C) -c
CXXLFLAGS= -L. -L$(LIBDIR) -s -ffast-math -L$(COMLIB)

ARCREATE= ar crv

include $(LEVEL2_SRC)/config/rules.common
```

These files simply specify the general rules for compilation. Specific rules for each S/W module are stated in the `planck.make` file in each module, or group of modules, directories.

In the following we will describe each module, or group of modules, in the same way they are orginalized in the Level 2 CVS repository of the LFI DPC.

5.1 CALIB

This set of modules are used for the photometric calibration of the difference LFI TOI. Their intent is a pure photometric calibration while removal of any systematic effects (e.g. 1/f noise stripes) are expected to be performed on calibrated TOI by map-making algorithms. The main members of this package are: `calib`, `newcalib` and `newcalib_iter`.

Module Name:	calib	Version:1.0	Source:	calib.f90
			Executable:	calib
Module Description:	This code implements the photometric calibration with CMB dipole using the independent set of differences of opposite points in a given scan circle. This is a parallel code parallelized with `MPI`.			
Build instructions:	make calib.clean, make calib.all; see planck.make in calib folder.			
Test Data Files:				

I/O Parameters:	`focalplane_db (string):`
	`  input file containing the detector database`
	
	`det_id (string):`
	`  name of the detector`
	
	`nominal_pointing (bool):`
	`  determines whether nominal (idealized) pointing is used`
	
	`levels_pointing (bool):`
	`  determines whether Level S (detector) pointing is used`
	
	`sat_info (string):`
	`  Name of the input file containing the satellite information`
	
	`first_pointing (int):`
	`  number of the first pointing period considered for`
	`  calibration. Default is 1.`
	
	`last_pointing (int):`
	`  number of the last pointing period considered`
	
	`pointing_per_file (int):`
	`  number of pointing periods per file (in case detector pointing`
	`  are used)`
	
	`t_int_min (double):`			

	time interval (in minutes) for photometric calibration do_cut (bool): determines whether sky cut has to be performed on data before calibration is computed galcut_angle(double): value of the galactic latitude cut (degress). Symmetric cut is assumed along the galactic plane do_weight (bool): determine whether to weight data according to their noise properties and sky model alpha (double): parameter for the determination of data weights do_mask (bool): determines whether to use a user defined sky mask tod_file (string): file containing data to be calibrated (toi.LS_toi) tod_cal_file (string): file with calibration reference signal (toi.LS_toi) have_tod_cal (bool): determines whether calibration reference signal is supplied detpnt_file (string): file with detector pointing information map_mask_file (string): file with user supplied sky mask map_weight_file (string): file with pixel weight mask g_file (string): output file with calibration constant values (calib.LFI_Gain_table) n_side (int): healpix nside parameter for input map
ApplicableURDs :	[AD-7]
SPRs/RIDs:	

Module Name:	newcalib	Version:1.0	Source:	newcalib.cc
			Executable:	newcalib
Module	This code performes photometric calibration using CMB dipole and all the samples on each			

Description:	scan circles (in opposite to the differences taken by the `calib` module).
Build instructions:	make calib.clean, make calib.all; see planck.make in calib folder
Test Data Files:	

I/O Parameters:	`first_pointing (int):`
	`  number of the first pointing period considered for`
	`  calibration. Default is 1.`
	
	`last_pointing (int):`
	`  number of the last pointing period considered`
	
	`input_TOI (string):`
	`  file containing data to be calibrated (toi.LS_toi)`
	
	`calib_TOI (string):`
	`  file with calibration reference signal (toi.LS_toi)`
	
	`out_cal_toi (string):`
	`  file with calibrated data (toi.LS_toi)`
	
	`detpnt (string):`
	`  file with detector pointing information`
	
	`input_map (string):`
	`  optional file with input signal map`
	
	`maskfile (string):`
	`  file with user supplied sky mask`
	
	`g_file (string):`
	`  output file with calibration constant values`
	`(calib.LFI_Gain_table)`
	
	`nbins (int):`
	`  number of samples along a scan ring`

ApplicableURDs :	[AD-7]
SPRs/RIDs:	

Module Name:	newcalib_iter	Version:1.0	Source:	newcalib_iter.cc
			Executable:	newcalib_iter
Module Description:	This code performs photometric calibration using all the samples in each scan ring comparing the signal with the CMB dipole. The code is iterative in the sense that previous calibration solultion is used to create a sky map less dipole. This is then scanned according to the pointing list and the corresponding TOI is subtracted from the original one resulting in a dipole dominated TOI. This in general improve accuracy in the calibration.			
Build instructions:	make calib.clean, make calib.all; see planck.make in calib folder			

Test Data Files:	
I/O Parameters:	first_pointing (int): number of the first pointing period considered for calibration. Default is 1. last_pointing (int): number of the last pointing period considered input_TOI (string): file containing data to be calibrated (toi.LS_toi) calib_TOI (string): file with calibration reference signal (toi.LS_toi) out_cal_toi (string): file with calibrated data detpnt (string): file with detector pointing information input_map (string): optional file with input signal map maskfile (string): file with user supplied sky mask g_file (string): output file with calibration constant values (calib.LFI_Gain_table) nbins (int): number of samples along a scan ring niter (int): number of iterations for signal subtraction
ApplicableURDs :	[AD-7]
SPRs/RIDs:	

5.2 CXXTOOLS

This is a collection of tools, mainly in C++, that are used during processing of the data.

Module Name:	applyG	Version:1.0	Source:	applyG.cc
			Executable:	applyG
Module Description:	This code takes the calibration constants computed by the photometric calibration module and apply them to the raw data to have data calibrated in Kelvin. This is a pure photometric			

	calibration. Calibrated data means simply converted from Volts to Kelvins.
Build instructions:	Make cxxtools.clean, make cxxtools.all; see planck.make in cxxtools folder.
Test Data Files:	

I/O Parameters:	`toi_file (string)`
	`  input LFI TOI (toi.science.LFI_DataDiff) to be calibrated`
	`  (i.e. converted into Kelvins)`
	
	`g_file (string)`
	`  input calibration table (calib.LFI_Gain_table)`
	
	`dipole_file (string)`
	`  reference dipole signal to be subtracted from the data. It is`
	`  expected to be sampled at the same rate of the input TOI`
	`  (toi.LS_toi)`
	
	`outfile (string)`
	`  output LFI TOI (toi.science.LFI_DataDiff) calibrated dipole`
	`  subtracted`
	
	`nominal_pointing (bool)`
	`  determines whether nominal (idealized) pointing is used`
	
	`detector_id (string)`
	`  name of the detector`
	
	`focalplane_db (string)`
	`  input file containing the detector database`

ApplicableURDs :	[AD-7]
SPRs/RIDs:	

Module Name:	CoaddTOI	Version:1.0	Source:	coaddTOI.cc
			Executable:	coaddTOI
Module Description:	This code implements simple algebric operations with TOI. They can be multiplied by user specified factors and the coadded together. This is used in the noise pipeline when an estimate of the signal is subctracted from the original signal+noise TOI.			
Build instructions:	Make cxxtools.clean, make cxxtools.all; see planck.make in cxxtools folder.			
Test Data Files:				

I/O Parameters:	`ntoi (int)`
	`  number of TOI to be coadded`
	
	`toi# (string)`
	`  input file with toi to be coadded (toi.science.LFI_DataDiff)`
	
	`factor# (double)`
	`  factors that TOI are multiplied by`			

	`outfile (string)` `  output coadded TOI (toi.science.LFI_DataDiff)`
ApplicableURDs :	[AD-7]
SPRs/RIDs:	

Module Name:	makeflags_cxx	Version:1.0	Source:	makeflags_cxx.cc
			Executable:	makeflags_cxx
Module Description:	This code creates a comprehensive flag from all input data to a given module.			
Build instructions:	Make cxxtools.clean, make cxxtools.all; see planck.make in cxxtools folder.			
Test Data Files:				

I/O Parameters:	`nominal_pointing (bool)`
	`  determines whether nominal (idealized) pointing is used`
	
	`nbins (int)`
	`  number of samples in a ring (1 minute)`
	
	`nrings (int)`
	`  total number of rings to be considered`
	
	`nflags_file (int)`
	`  number of flags file to be read (then logically merged)`
	
	`detector_id (string)`
	`  name of the detector`
	
	`focalplane_db (string)`
	`  input file with detector database`
	
	`file_flag# (string)`
	`  input file with flags (flag.LFI_flag)`
	
	`file_oflags (string)`
	`  output file with logically merged flags (flag.LFI_flag)`			

ApplicableURDs :	[AD-7]			
SPRs/RIDs:				

Module Name:	map2TOD	Version:1.0	Source:	map2TOD.cc
			Executable:	map2TOD
Module Description:	This code takes pointing information for a given detector and from an input map, creates the corresponding TOI.			
Build instructions:	Make cxxtools.clean, make cxxtools.all; see planck.make in cxxtools folder.			

Test Data Files:	

I/O Parameters:	`file_tod (string)`
	`  output TOD file from scanned map (toi.science.LFI_DataDiff)`
	
	`file_pnt (string)`
	`  input detector pointing information`
	`(pointing.LS_detpoint_with_orientation)`
	
	`input_map (string)`
	`  map to be scanned to produce TOD`

ApplicableURDs :	[AD-7]
SPRs/RIDs:	

Module Name:	phasebin(_LS2LS, _LFI2LS)	Version:1.0	Source:	phasebin_(LS2LS, _LFI2LS).cc
			Executable:	phasebin_(LS2LS, LFI2LS)
Module Description:	This code takes input TOI and pointing information (in both Level S and LFI data type format) and creates phase bin rings with Level S data type.			
Build instructions:	Make cxxtools.clean, make cxxtools.all; see planck.make in cxxtools folder.			
Test Data Files:				

I/O Parameters:	`nominal_pointing (bool)`
	`  determines whether nominal (idealized) pointing is used`
	
	`nbins (int)`
	`  number of sample in a ring`
	
	`detector_id (string)`
	`  name of the detector`
	
	`focalplane_db (string)`
	`  input file with detector database`
	
	`first_pointing (int)`
	`  first pointing to be considered`
	
	`last_pointing (int)`
	`  last pointing to be considered`
	
	`omega_spin (double)`
	`  angular S/C velocity`
	
	`domega_spin (double)`
	`  angular S/C accelleration`
	
	`tod_filename (string)`
	`  input TOI to be phase binned (toi.science.LFI_DataDiff)`			

	pnt_filename (string)
	  input detpnt to be phase binned
	(pointing.LS_detpoint_with_orientation_dp)
	
	todbin (string)
	  output phase binned tod (toi.LS_toi)
	
	pntbin (string)
	  output phase binned detector pointing
	(pointing.LS_detpoint_with_orientation_dp)

ApplicableURDs :	[AD-7]
SPRs/RIDs:	

Module Name:	removeBaseline	Version:1.0	Source:	removeBasline.cc
			Executable:	removeBaseline
Module Description:	This code removes baselines obtained with madam map-making code and subctrat them from raw calibrated TOI. This is useful for further processing of calibrated clean TOI.			
Build instructions:	Make cxxtools.clean, make cxxtools.all; see planck.make in cxxtools folder.			
Test Data Files:				

I/O Parameters:	detector_id (string)
	  name of the detector
	
	focalplane_db (string)
	  input file with detector database
	
	toi_file (string)
	  input toi to be baseline subtracted (toi.science.LFI_DataDiff)
	
	base_file (string)
	  input baseline file
	
	base1h (bool)
	  determines whether baselines are 1 hour based or smaller
	
	if (!base1h)
	  base_length (int)
	  baseline length in units of samples
	endif
	
	nominal_pointing (bool)
	  determined whether nominal (idealized) pointing is used
	
	outfile (string)
	  output baseline subtracted TOI (toi.science.LFI_DataDiff)			

ApplicableURDs :	[AD-7]			
SPRs/RIDs:				

Module Name:	r_param_SOVT1	Version:1.0	Source:	r_param_SOVT1.cc
			Executable:	r_param_SOVT1
Module Description:	This code takes the un-differenced LFI data, compute the gain modulation factor R and create differenced data. This version is the one used for the SOVT1 tests. It reads data directly from the LEVEL1 DB and stores outputs in the LEVEL2/LEVEL3 DB.			
Build instructions:	Make cxxtools.clean, make cxxtools.all; see planck.make in cxxtools folder.			
Test Data Files:				

I/O Parameters:	`Oday (int)`
	`  operational day to be considered`
	
	`detector_id (string)`
	`  name of the detector`
	
	`output_DB (string)`
	`  name of the output DataBase`			

ApplicableURDs :	[AD-7]			
SPRs/RIDs:				

5.3 FILTERS

This is a collection of tools like high pass filter and two independent algorithms for cleaning data from frequency spikes.

Module Name:	fitspike	Version:1.0	Source:	Fitspike.f90
			Executable:	fitspike
Module Description:	This code implements a simple fitting procedure for frequency spikes (due to H/K acquisition interference) in time domain			
Build Instructions:	Make filters.clean, make filters.all; see planck.make in filters folder			
Test Data Files:				

I/O Parameters:	`focalplane_db (string)`
	`  input file with detector database`
	
	`detector_id (string)`
	`  name of the detector`
	
	`spike_file (string)`
	`  file with known spike positions (limo.Spikes)`
	
	`input_file (string)`
	`  file with LFI differenced data to be cleaned`			

	(toi.science.LFI_DataDiff) outfile (string) output file with spikes removes (toi.science.LFI_DataDiff) spike_mod (bool) whether to define spikes position manually or read from the limo.Spikes file n_hours (int) number of hours to be processed at a given time
ApplicableURDs :	[AD-7]
SPRs/RIDs:	

Module Name:	remove_spikes	Version:1.0	Source:	remove_spikes.cc
			Executable:	remove_spikes
Module Description:	This codeimplements the removal of frequency spikes directly in frequency domain for those spikes whose position is known from Ground test data analysis.			
Build Instructions:	make filters.clean, make filters.all; see planck.make in filters folder.			
Test Data Files:				
I/O Parameters:	detector_id (string) name of the detector spikes (string) input file with known spikes positions time_domain (bool, default=false) determines whether to perform a time or a Fourier domain cleaning tod (string) input tod to be cleaned (toi.science.LFI_DataDiff) cleanedtod (string) output cleaned tod (toi.science.LFI_DataDiff)			
ApplicableURDs :	[AD-7]			
SPRs/RIDs:				

Module Name:	High-pass filter	Version:1.1	Source:	high_pass.cc
			Executable:	highpass
Module Description:	This code performs an high-pass filter on TOD for systematic effect study. It has been tested with both 1/f noise and thermal fluctuations induced by the sorption cooler and transferred from the cold-end to the output of the radiometer.			

Build Instructions:	make filters.clean, make filters.all; see planck.make in filters folder
Test Data Files:	

I/O Parameters:	`store_dir (string)`
	`  location of output files`
	
	`det_id (string)`
	`  name of the detector`
	
	`focalplane_db (string)`
	`  input file with detector database`
	
	`filter_cutoff (double)`
	`  filter cutoff frequency`
	
	`retain_baselines (bool)`
	`  determines whether baselines from filter application has to be retained`
	
	`filter_slope (int)`
	`  filter slope`
	
	`filter_type (int)`
	`  type of filter used: 0 - for BW filter, 1 - for step function`
	
	`file_tod (string)`
	`  input TOD to be filtered (toi.LS_toi)`
	
	`outfile (string)`
	`  output filtered tod (toi.LS_toi)`

Applicable URDs :	[AD-7]
SPRs/RIDs:	

5.4 FITTERS

This is a collection of tools for the beam reconstruction using planet transit.

Module Name:	beamfits	Version:1.0	Source:	beamfits.cc
			Executable:	beamfits
Module Description:	This module implements a bi-variate gaussian fitting of data from planet transit to get the beam parameters			
Build Instructions:	make fitters.clean, make fitters.clean; see planck.make in fitters folder			
Test Data Files:				
I/O Parameters:	`alpha (double)`			

```
  initial value of beam orientation

s_alpha (double)
  initial guess of beam orientation accuracy

x0 (double)
  initial value of beam x-position on the focal plane

s_x0 (double)
  initial guess of beam x-position on the focal plane

x0_min (double)
  lower limit for searching chi2 minimum

x0_max (double)
  upper limit for searching chi2 minimum

y0 (double)
  initial value of beam y-position on the focal plane

s_y0 (double)
  initial guess of beam y-position on the focal plane

y0_min (double)
  lower limit for searching chi2 minimum

y0_max (double)
  upper limit for searching chi2 minimum

sigma (double)
  initial value of beam sigma

s_sigma (double)
  initial guess of beam sigma accuracy


r_min (double)
  lower limit for searching chi2 minimum

r_max (double)
  upper limit for searching chi2 minimum

rk (double)
  initial value of beam peak signal

s_rk (double)
  initial guess of beam peak signal accuracy

rk_min (double)
  lower limit for searching chi2 minimum

rk_max (double)
  upper limit for searhcing chi2 minimum
```

	noise_sample_variance (double)
	  noise variance per sample
	
	sigma_clipping (double)
	  value of sigma clipping treshold
	
	outfile (string)
	  output file with beam fit results (beam_fit.Output)
	
	xy_file (string)
	  input file with x,y, distance and signal of planet crossing on the
	  focal plane (xy.LFI_beam_xy)

ApplicableURDs :	[AD-7]
SPRs/RIDs:	

Module Name:	getXYplanet	Version 1.0	Source:	getXYplanet.cc
			Executable	getXYplanet
Module Description	This code implements the computation of the X and Y position of planet transit on the focal plane. This is then used by the beamfit code for getting the beam parameters.			
Build Instructions:	make fitters.clean, make fitters.all; see planck.make in fitters folder.			
Test Data Files:				

I/O Parameters:	detector_id (string)
	  name of the detector
	
	focalplane_db (string)
	  input file with detector data base
	
	nominal_pointing (bool)
	  whether nominal (idealized) pointing is used
	
	obs_radius (double)
	  dimensions in degrees of the searching region for planets
	
	planet_id (int)
	  name of the planet according to simmission convention
	
	start_pnt_analysis (int)
	  starting pointing id for the analysis
	
	end_pnt_analysis (int)
	  ending pointing id for the analysis
	
	solsys (string)
	  input file with planets positions (catalog.LS_planet_data)
	
	toi (string)			

	input file with data to be searched for xy_file (string) output file with x,y,d and signal of planet crossing (xy.LFI_beam_xy)
Applicable URDs:	[AD-7]
SPRs/RIDs	

5.5 GDESTRI_CXX

This module is a generalized destriper where base fitting functions are not simply offsets but are allowed to be Legendre polynomials as well ass Fourier modes.

Module Name:	gdestri_cxx	Version:2.1	Source:	gdestri_cxx.cc
			Executable:	gdestri
Module Description:	This code is an evolution of the classic destriping code in the sense that allows to fit the systematic effects on Averaged Rings with different base functions (simple baselines, Fourier modes, Legendre polynomials) thus picking-up different aspects of the systematic effects. This is a serial code and works only in temperature.			
Build Instructions:	make gdestri_cxx.clean, make gdestri_cxx.all; see planck.make in gdestri_cxx folder			
Test Data Files:				
I/O Parameters:	focalplane_db (string) input file with detector data base detector_id (string) name of the detector nside_map (int) Healpix map resolution parameter for output map nside_pair (int) Healpix map parameter for crossing between scan rings nrings (int) number of rings to be considered			

	nbins (int)
	  number of samples along a ring
	
	nofit (int)
	  number of fitting functions
	
	fitmod (int)
	  type of fitting mode (1 sine/cosine, 2 polynomials)
	
	file_tod (string)
	  input TOD file (toi.LS_toi)
	
	file_pix (string)
	  file with detector pointing information
	
	file_hit (string)
	  output map with hits per pixel
	
	file_raw (string)
	  output raw (binned) map
	
	file_map (string)
	  output destriped map
	
	outtod (string, optional)
	  output cleaned TOD (toi.LS_toi)

Applicable URDs :	[AD-7]
SPRs/RIDs:	

5.6 MADAM

This is an hybrid map-making algorithm that can be a destriped with different base lenghts as well as an IGLS map-making when TOI filtering is allowed. This is a parallel code and work both in temperature and polarisation.

Module Name:	madam	Version:3.0	Source:	Smadam.f90
			Executable:	smadam
Module Description:	This code implements an hybrid approach to map-making. This can be a destriping with variable baseline length as well as an optimal map-making.			
Build Instructions:	Make madam3.0.clean, make madam3.0.all; see planck.make in madam3.0 folder.			
Test Data Files:				
I/O Parameters:	focalplane_db (string) input file with detector database			

```
detector_# (string)
  name of the #th detector (several can be provided)

file_point_# (string)
  input file with pointing information of the #th detector

file_tod_# (string)
  input file with TOD of the #th detector

file_map (string)
  output cleaned map

file_mask (string)
  outmap mask map (optional)

file_matrix (string)
  output pixel matrix map (optional)

file_binmap (string)
  output binned map (optional)

file_inmask(string)
  input mask file

filter_rings (int, default=60)
  length of noise filter as number of rings

fsample (double)
  sampling frequency in Hz

info (int)
  verbosity of the code
interp_mode (int, default=4)
  type of TOD/pointing interpolations correction for pointing
shift:
  0 - no interpolation, 1 - TOD interpolations, 2 - Pointing
  interpolations, 3 - as in 2 but no interpolations from ring to
ring,
  4 - as 2 but first sample in each ring found by interpolation

intstep (int, default=0)
  interpolation step

isigma_base (double, default=0)
  a priori inverse STD for the baselines in units of 1/sigma

isigma_map (double, default=0)
  a priori inverse STD of the signal component

kprecond (bool, default=true)
  if true improve CG convergence

kfirst (bool, default=true)
```

```
  whether to perform first destriping

ksecond (bool, default=false)
  whether to perform second destriping

kfilter (bool, default=false)
  whether to use noise filter. if false madam is a classical
destriper

mission_time (int, default=0)
  mission time in time_units

nosamples (int)
  number of samples in a ring

noloops_point (int, default=1)
  this controls the division of concurrent timelines in split-
mode

noloops_time (int, default=1)
  this controls the division of a timeline in split-mode
nobuffer (int, default=1)
  size of the TOD read-in buffer

base_first (int)
  baseline length in the first destriping phase (units of
samples)
  relevant only if kfirst = true

base_second (int, default=60)
  baseline length in the second destriping as a number of rings
  relevant only if ksecond = true

cglimit (double)
  criterium for Conjugate Gradient solving

nside_cross (int, default=512)
  healpix resolution for destriping

nside_map (int, default=512)
  healpix resolution output map

nside_submap (int, default=16)
  size of submap which is a basic data unit for MPI
communication

pixlim_cross (double, default=1.e-6)
  smallest accepted value for the quality selected by
pixmode_cross

pixlim_map (double, default=1.e-6)
  smallest accepted value for quality selected by pixmode_map

pixmode_cross (int, default=2)
```

	criterion for discarding degenerate pixels 0 - absolute determinant of pixel matrix 1 - scaled determinant (by the trace) 2 - reciprocal condition number 4 - matrix inversion pixmode_map (int, default=2) criterion for discarding degenerate pixels 0 - absolute determinant of pixel matrix 1 - scaled determinant (by the trace) 2 - reciprocal condition number 4 - matrix inversion pixmode_map (int, default=2) criterion for discarding degenerate pixels in output map rm_monopole (bool, default=false) whether to subtract monopole start_time (int, default=0) starting point in time_units tail_rings (int , default=10) length of overlapping data section as number of rings time_unit (int, default=60) arbitrary unit as number of rings tod_# (string) input file with TOD from the #th detector
Applicable URDs :	[AD-7]
SPRs/RIDs:	

5.7 NOISE TOOLS

This is a collection of tools useful for estimation of noise parameters from LFI data. These are white noise sensitivity, knee-frequency and slope of the 1/f noise. There are two fitting procedures fitnoise_GT and whittle. The first one is a simple log-periodogram algorithm where a linear fit in log-log is performed to get slope and then knee-frequency. The whille algorithm implements the Whittle estimate which is a maximum-likelihood estimate.

Module Name:	fitnoise_GT	Version:1.0	Source:	fitnoise_GT.cc
			Executable:	fitnoise_GT
Module Description:	This file implements a fit procedure for noise power spectrum according to the code used for LFI Ground Tests. This returns fk, slope and white noise level.			
Build Instructions:	make noise_tools.clean, make noise_tools.all; see planck.make in noise_tools folder.			

Test Data Files:	

I/O Parameters:	`ndata_fk (int)`
	`  number of frequency bins (from first_valid_mode) used for computing`
	`  slope and knee-frequency`
	
	`first_valid_mode (int)`
	`  first valid mode to be considered in the noise parameter fit`
	
	`psfile (string)`
	`  input power spectrum (noise_ps.LFI_noise_ps)`
	
	`noise_params (string)`
	`  output fitted noise parameters (noise_ps.LFI_parameters)`

ApplicableURDs :	[AD-7]
SPRs/RIDs:	

Module Name:	whittle	Version:1.0	Source:	Whittle.f90
			Executable:	whittle
Module Description:	This code implements the whittle estimation of the noise parameters.			
Build Instructions:	make noise_tools.clean, make noise_tools.all; see planck.make in noise_tools folder.			
Test Data Files:				

I/O Parameters:	`ps_file (string)`
	`  input power spectrum (noise_ps.LFI_noise_ps)`
	
	`noise_params (string)`
	`  output fitted noise parameters (noise_ps.LFI_parameters)`
	
	`first_valid_mode (int)`
	`  first valid mode to be considered in the fit`
	
	`guess_wn (double)`
	`  guess of the white noise level ([TOI]^2)`
	
	`guess_fk (double)`
	`  guess value for the knee frequency [Hz]`
	
	`guess_alpha (double)`
	`  guess value for the spectrum slope`			

ApplicableURDs :	[AD-7]			
SPRs/RIDs:				

Module Name:	noise_ps	Version 1.0	Source:	noise_ps.cc
			Executable	noise_ps

Module Description	This code implements the computation of the fourier noise spectrum for a given TOI.
Build Instructions:	make noise_tools.clean, make noise_tools.all; see planck.make in noise_tools folder.
Test Data Files:	

I/O Parameters:	`detector_id (string)`
	`  name of the detector`
	
	`focalplane_db (string)`
	`  input file with detector data base`
	
	`infile (string)`
	`  file name with input toi (toi.science.LFI_DataDiff)`
	
	`fast (loop)`
	`  whether power of two data has to be used`
	
	`out_hour (string)`
	`  output file with hour power spectra (noise_ps.LFI_noise_ps)`
	
	`out_day (string)`
	`  output file with full day power spectrum (noise_ps.LFI_noise_ps)`
	
	`out_have (string)`
	`  output file with day spectra from average of 24 hourly ones (noise_ps.LFI_nois e_ps)`

Applicable URDs:	[AD-7]
SPRs/RIDs	

5.8 PNTSRC

This module implements the point source extraction from LFI calibrated cleaned TOI. This is based on wavelets as well as on optimal pseudo-filters.

Module Name:	pntsrc	Version:1.0	Source:	TOD.f90
			Executable:	SExtra
Module Description:	This code perfomes a point source extraction on TOD using different methods (wavelets, optimal pseudo-filters) and returns cleaned TOD and a list of possible detected point sources at different values of accuracy (threshold with respect white noise level).			
Build Instructions:	make pntsrc.clean, make pntsrc.all; see planck.make in pntsrc folder.			
Test Data Files:				

I/O Parameters:	input_tod (string)
	  input file with tod to be processed
	
	input_detpnt (string)
	  input file with detector pointing information
	
	output_tod (string)
	  output filtered TOD
	
	n_rings (int)
	  number of rings in the TOD
	
	n_pixels_ring (int)
	  number of samples in a ring
	
	pixesize_arcmin (double)
	  pixel size in arcmin
	
	antenna_fwhm (double)
	  antenna FWHM in arcmin
	
	n_averings (int)
	  number of averaged rings in the analysis
	
	n_avespec (int)
	  number of averaged power spectra in the analysis
	
	n_cnctpix (int)
	  number of connected pixels around a maximum
	
	filter (int)
	  filter type
	  1 - Gaussian
	  2 - Mexican Hat
	  3 - Optimal Pseudo-Filter
	  4 - Matched Filter
	
	scale_width (double)
	  filter scale in arcmin

Applicable URDs :	[AD-7]
SPRs/RIDs:	

5.9 QUATERNION

This is a set of tools which handles quaterions. There are two codes which converts quaterion in both Level S format (output from multimod module) as well as the standard AHF provided by MOC, into detector pointings. C++ and F90 libraries for handling quanternions are also provided.

Module Name:	AHF2detpnt	Version 1.0	Source:	AHF2detpnt.cc
			Executable	AHF2detpnt
Module Description	This code implements the conversions of pointing information encoded into quaternions as stored into the AHF file, into detector pointing information.			
Build Instructions:	make quaterion.clean, make quaternion.all; see planck.make in quaternion folder.			
Test Data Files:				

I/O Parameters:	`focalplane_db (string)`
	`  input file with detector database`
	
	`detector_id (string)`
	`  name of the detector`
	
	`inAHF (string)`
	`  input file with AHF file (toi.attitude.HighFrequency)`
	
	`inTOI (string)`
	`  input file with TOI for getting time of samples (toi.science.LFI_DataDiff)`
	
	`outfile (string)`
	`  output file with detector pointing (pointing.LS_detpnt_with_orientation)`
	
	`single_precision_pointing (bool,default=false)`
	`  whether to store pointing in double or single precision`			

Applicable URDs:	[AD-7]			
SPRs/RIDs				

Module Name:	quat2detpnt	Version 1.0	Source:	quat2detpnt.cc
			Executable	quat2detpnt
Module Description	This code implements the conversions of pointing information encoded into quaternions as stored into the quaternion file produced by multimod module of Level S, into detector pointing information.			
Build Instructions:	make quaterion.clean, make quaternion.all; see planck.make in quaternion folder.			
Test Data Files:				

I/O Parameters:	`focalplane_db (string)`
	`  input file with detector database`
	
	`detector_id (string)`
	`  name of the detector`
	
	`infile (string)`
	`  input file with quaternion(quat.LS_satpt_quat)`
	
	`outfile (string)`
	`  output file with detector pointing`
	`(pointing.LS_detpnt_with_orientation)`

Applicable URDs:	[AD-7]
SPRs/RIDs	

5.10 IGLS MAP-MAKING

Module Name:	Roma and Romapol	Version:1.2	Source:	Roma.f90, romapol.f90
			Executable:	roma, romapol
Module Description:	This is an iterative general least square optimal map-making developed by the Rome group. Roma stands for: Roma Optimal Map-making Algorithm. It is a parallel algorithm and works on both temperature and polarisation data.			
Build Instructions:	Make roma.clean, make roma.all; see planck.make in roma folder.			
Test Data Files:				

I/O Parameters:	`mm_pointing_file (string)`
	`  file with detector pointing information (both theta and phi and orientation psi)`
	
	`mm_filter_file (string)`
	`  file with noise filter (derived from data themselves)`
	
	`mm_xpol_file (string)`
	`  file with cross-polarisation information`
	
	`mm_tod_file#horn (string)`
	`  collection of TOI for the specified number of horns to be processed`
	
	`mm_binned_map (string)`
	`  binned output map`
	
	`mm_white_map (string)`
	`  output whitened map`			

	mm_hits_map (string) map of hits per pixel mm_diag_map (string) diagonal noise matrix mm_igls_map (string) output IGLS map mm_nhorns (int) number of horn to be processed mm_do_interesaction (bool, default=false) do analysis on common horn area mm_do_rhs (bool, default = true) solve RHS of the MM equation mm_do_lhs (bool, default = true) solve LHS of the MM equation mm_ndata (int) number of samples in each TOI mm_filter_len (int) filter length (in samples) mm_nside (int) nside resolution parameter for output maps mm_accuracy (double) Conjugate Gradient accuracy mm_max_iters (int) maximum number of CG iterations
Applicable URDs :	[AD-7]
SPRs/RIDs:	

5.11 FASTICA

This module impements the component separation by means of the Independent Component Analysis technique.

Module Name:	FastICA	Version:1.1	Source:	fastica.f90
			Executable:	fastica

Module Description:	This is a code based on Independent Component Analysis (ICA) aiming at the extraction of the astrophysical component in the observed signal. It is based on a measure of non-gaussianity called Neg-Entropy and assumes that astrophysical components are statistical independent and with no correlation. This has been proved to work on Planck like simulations as well as on real data (COBE-DMR, WMAP). A parallel version (with `MPI`) is implemented with optional control on the CMB frequency scaling as well as data compression through PCA analysis.
Build Instructions:	make fastica.clean, make fastica.all; see planck.make in fastica folder.
Test Data Files:	

I/O Parameters:	`file# (string)`
	`  files with input HEALPix maps from different frequency channels`
	
	`fileQ(U)# (string)`
	`  files with input Q and U HEALPix maps from different channels`
	
	`rmsX# (string)`
	`  files with (X=I,Q,U) rms for different frequency channels`
	
	`noiseX# (string)`
	`  files with (X=I,Q,U) noise maps for different frequency channels`
	
	`u# (string)`
	`  output map with extracted independent components`
	
	`a_table (string)`
	`  file with inverse weighting matrix`
	
	`w_table (string)`
	`  file with weighting matrix`
	
	`icatype (int)`
	`  type of ica separation (T = 1, QU = 2)`
	
	`qutype (int)`
	`  type of QU separation (1 = QU sep, 2 = QU together)`
	
	`nsig (int)`
	`  number of input signals`
	
	`n_iter(int)`
	`  number of ICA iterations`
	
	`epsilon (double)`
	`  relative ICA accuracy`
	
	`a1 (double)`
	`  first parameter for one of the ICA functions`
	
	`a2 (double)`
	`  second parameter for another ICA function`

	non_lin (string) type of non-quadratic approximation of Neg-Entropy oknoise (bool, default=false) have noise maps okrms (bool, default=false) have rms maps regression (int) whether to remove residual monopole, dipole from maps noise_in_data (bool, default=false) whether data contains noise or not cure_noise (int) type of noise cure (with rms, with noise maps) cut (int) type of galactic cut (full-sky, iso-latitude, mask)
Applicable URDs :	[AD-7]
SPRs/RIDs:	

5.12 CCA (CORRELATED COMPONENT ANALYSIS)

This code implements the Correlated component analysis that is used for component separation and to gain information on the spectral properties of the component signals.

5.13 POINT SOURCE AND CLUSTER EXTRACTION

5.13.1 MULTIFILTER

Module Name:	MultifilterPSD	Version:1.1	Source:	mathced_filter.f90
			Executable:	matched_filter
Module Description:	This code implement point source detection from full-sky maps (which are divided into patches) with different approaches: the first two members of the Mexican Hat wavelet family and matched filter.			
Build Instructions:	make MultifilterPSD.clean, make MutlifilterPSD.all; see planck.make in MultifilterPSD folder			

Test Data Files:	
I/O Parameters:	deltatheta_deg (double) size of single patch [deg] theta_overlap (double) size of the overlap between patches [deg] FWHM (double) antenna FWHM [arcmin] matching_radius (double) matching radius to remove repeated sources [arcmin] border_edge (double) size of the patch border not to be considered chi2_limit (double) limit of chi2 during the fit detection_threshold (double) threshold in sigma in the filtered image image_size_pixels (int) image size in pixels input_map (string) input map output_catalogue (string) output catalogue filter (int) type of filter used (1:MHW1, 2:MHW2, 3:MF) calibration_file (string) calibration file (if any) diagnostics (bool, default=false) run in diagnostic mode write_patches (bool, default = false) write patches and stop connected_pixels (int) number of connected pixels to define a maximum sigma_estimation (int) type of sigma estimation: 1=conservative, 2=improved
Applicable URDs :	[AD-7]
SPRs/RIDs:	

5.13.2 SZ_MATCHEDMULTIFILTER

Module Name:	SZ_MatchedMultiFilter	Version:1.1	Source:	SZclustersdet.f90
			Executable:	SZclustersdet
Module Description:	This code performs a specific extraction of Compton y parameter from SZ observation. It works with several multi-frequency input data.			
Build Instructions:	make SZ_MathcedMultiFilter.clean, make SZ_MatchedMultiFilter.all; see planck.make in SZ_MatchedMultiFilter folder.			
Test Data Files:				

I/O Parameters:	`deltatheta_deg (double)`
	`  size of single patch [deg]`
	
	`theta_overlap (double)`
	`  size of the overlap between patches [deg]`
	
	`detection_threshold (double)`
	`  threshold in sigma in the filtered image`
	
	`image_size_pixels (int)`
	`  image size in pixels`
	
	`input_map_# (string)`
	`  a given number of multi-frequency input patches (usually 9)`
	
	`output_catalogue (string)`
	`  output catalogue`
	
	`number_iterations (int)`
	`  total number of iterations`			

Applicable URDs :	[AD-7]			
SPRs/RIDs:				

5.13.3 POWELLSNAKE

Module Name:	PwSnake	Version:2.0	Source:	PowellSnakes.cpp
			Executable:	PwSnake
Module Description:	PowellSnake implements fast Bayesian approach for the detection of discrete objects immersed in a diffuse background. Gaussian shape for the peaks (I.e. sources) is			

	assumed.
Build Instructions:	Make PwSnake.clean, make PwSnake.all; see planck.make in PwSnake folder
Test Data Files:	
I/O Parameters:	mapreader_reader (string) kind of input map healpix_map_zones (string) string with digits : separated with position of the sky region of interest mapreader_file_name (string) input map (map.LS_map) healpix_use_map_defs (int, default=1) definition of usage for input map healpix_coors_sys (int, default=2) coordinate system of input map (2 = Galactic) healpix_map_class_coords_sys (int, default=2) coordinate sysyem of the zone map ptg000000_units (int, default=0) units of the input map (0 = K_antenna, 1 = K_thermo, 2 = Mjr/sr) post_processor (string) type of output catalogue pp_catalog_out_fname (string) filename of output catalogue pp_min_flux (double) minimum flux in the catalogue ptg000000_antenna_freq_00 (double) frequency of the input map ptg000000_antenna_gain_factor_00 (double) gain factor for input data ptg000000_antenna_inst_noise_00 (double) percentage of instrument noise contributing to total RMS
Applicable URDs :	[AD-7]
SPRs/RIDs:	

5.14 POWER SPECTRUM ESTIMATION

5.14.1 MASTER

Module Name:	MASTER Deconv	Version:1.1	Source:	deconv.f90
			Executable:	deconv
Module Description:	This code takes the results from MC simulation of pure noise (estimated from the data or from an instrument model), pure signal (effects of the scanning strategy and reduction approaches) plus the observed (from data) power spectrum on a given sky mask, combine them with a proper kernel for the sky area considered and returns the un-bias estimated true power spectrum by means of a MASTER like approach.			
Build Instructions:	make master.clean, make master.all; see planck.make in master folder.			
Test Data Files:				

I/O Parameters:	`kern_file (string)`
	`  input file with kernel`
	
	`cl_noise (string)`
	`  full set of noise only power spectra from MC simulations runs (powerspectrum.LS_powspec_pol)`
	
	`cl_observed (string)`
	`  observed angular power spectrum (powerspectrum.LS_powspec_pol)`
	
	`filter function (string)`
	`  filter function for pure signal MC depending on map-making procedured (powerspectrum.LS_powspec_pol)`
	
	`cl_bin_out (string)`
	`  output un-biased estimatior of CMB power spectrum binned in ell space (powerspectrum.LS_powspec_pol)`
	
	`nside (int)`
	`  Healpix resolution parameter`
	
	`bin_str (int)`
	`  number of bins and size`			

Applicable URDs :	[AD-7]			
SPRs/RIDs:				

Module Name:	Kern_gabor	Version:1.1	Source:	kern_gabor.f90
			Executable:	Kern
Module Description:	This code produce the Kernel matrix in multipole space needed by the MASTER approach to compute the power spectrum on a selected sky area. This is done to take into account the mode-mode coupling due to any incomplete sky coverage.			

Build Instructions:	Make master.clean, make master.all, see planck.make in master folder.
Test Data Files:	

I/O Parameters:	`cl_sky_area (string)`
	`  file with square of the window function for the considered area (powerspectrum.LS_powspec_pol)`
	
	`kern_file (string)`
	`  output file with coupling kernel`
	
	`lmax (int)`
	`  maximum multipole for kernel consturciton`
	
	`type (int)`
	`  kind of window (0=full-sky, 1=gaussian, 2=tophat, 3=user supplied)`
	
	`width (int)`
	`  width of the gaussian window [deg]`
	
	`radius (double)`
	`  radius of top hat function`

Applicable URDs :	[AD-7]
SPRs/RIDs:	

Module Name:	Sky Mask	Version:1.0	Source:	skyMask.f90
			Executable:	skyMask
Module Description:	This code produce a sky mask (with 0 and 1) according to user's specification. The area could be at a given latitude or longitude or the uses could supply verteces of the interested area			
Build Instructions:	make master.clean, make master.all; see planck.make in master folder.			
Test Data Files:				

I/O Parameters:	`mask_file (string)`
	`   output mask file (map.LS_map)`
	
	`scheme (string)`
	`  ordering of the Healpix pixelisation for output map`
	
	`respar (int)`
	`   Healpix resolution parameter`
	
	`isys (string)`
	`  input coordinate system of mask definition`
	
	`osys (string)`
	`  output coordinate of the mask`			

	cmod (string)
	  type of cut in co-latitude
	
	lmod (string)
	  type of cut in longitude
	
	lat_min (double)
	  minimum latitude of the cut
	
	lat_max (double)
	  maximum latitude of the cut
	
	lon_min (double)
	  minimum longitude of the cut
	
	lon_max (double)
	  maximum longitude of the cut
	
	alpha (double)
	  parameter for pixel weight

Applicable URDs :	[AD-7]
SPRs/RIDs:	

Module Name:	Apply Mask	Version:1.0	Source:	applyMask.f90
			Executable:	skyMask
Module Description:	This code applies a given sky mask to a given sky map and procude a masked sky map.			
Build Instructions:	make master.clean, make master.all; see planck.make in master folder.			
Test Data Files:				

I/O Parameters:	inmap_file (string)
	  input map to be maskter (map.LS_map)
	
	mask_file (string)
	  mask file to be used (map.LS_map)
	
	out_file (string)
	  output masked map (map.LS_map)			

Applicable URDs :	[AD-7]			
SPRs/RIDs:				

Module Name:	Xspect	Version:1.0	Source:	crossspect.cc
			Executable:	crossspect
Module	This code compute the temperature and polarisation cross-power spectrum from			

Description:	independent detectors assuming un-correlated noise. This is based on a MASTER approach i.e. it is not optimal for low ells. This does not requires noise only MC simulations for estimation of the power spectrum itself although it requires signal+noise power spectrum for proper error bars estimation.
Build Instructions:	make xspect.clean, make xspect.all; see planck.make in xspect folder.
Test Data Files:	

I/O Parameters:	`nlmax (int)`
	`  maximum multipole for the analysis`
	
	`nmmax (int, default=nlmax)`
	`  maximum m-mode for the analysis`
	
	`infile1 (string)`
	`  first input map (map.LS_map)`
	
	`infile2 (string)`
	`  second input map (map.LS_map)`
	
	`beam_file (bool, default=false)`
	`  whether have or not beam function`
	
	`beam1 (string)`
	`  input beam function for first map (bl.LFI_beamfunction)`
	
	`beam2 (string)`
	`  input beam function for second map (bl.LFI_beamfunction)`
	
	`fwhm1 (float)`
	`  fwhm of the first detector map`
	
	`fwhm2 (float)`
	`  fhwm of the second detector map`
	
	`polarisation (bool, default=false)`
	`  whether to perform polarisation analysis`
	
	`weighted (boo, default=false)`
	`  whether to use ring weights`
	
	`num_iter (int, default=0)`
	`  number of iterations in the analysis`
	
	`double_precision (bool, default=false)`
	`  whether to perform analysis in single or double precision`

Applicable URDs :	[AD-7]
SPRs/RIDs:	

Other power spectrum estimation tools are in the integration process after their successful porting on the DPC system. Among them there is ROMAster, a MASTER like approach auto- and cross- power spectrum estimation and BolPol which implements a quadratic maximum likelihood algorithm suitable for low (<60) multipoles. Maximum Nside is 16 since computational scaling make it unfeasible to be run the code at Nside > 32. Both codes are in the testing and benchmarking phase.

6 LEVEL 4 - CONFIGURATION LIST

TBW – No software delivery in Demonstration / Operation Model.

7 HARDWARE - CONFIGURATION LIST

The figure in the next page shows the design of the branch of the INAF-OATS LAN where the LFI DPC hardware is located, as defined in [RD-07]. From the logistical point of view, the figure identifies three separate areas, the LFI DPC building, the DPC Control Room and the DPC Computer Room.

For every hardware item, a full description is given in a separate form.

In addition to the hardware items specified below, networking has to be considered. The DPC uses two different dedicated branches of the OATs Local Area Network, one for SGS1 and one for SGS2, both supported by Fast Ethernet at 100 Mbps. The ENTMOOT cluster uses a private network based on Myrinet at 1.2 Gbps.

It is furthermore to be noted that the dedicated lines to the MOC (nominal and backup) and the relevant routers are shown in the figure for clarity but, being MOC's responsibility, are not under the DPC configuration control.

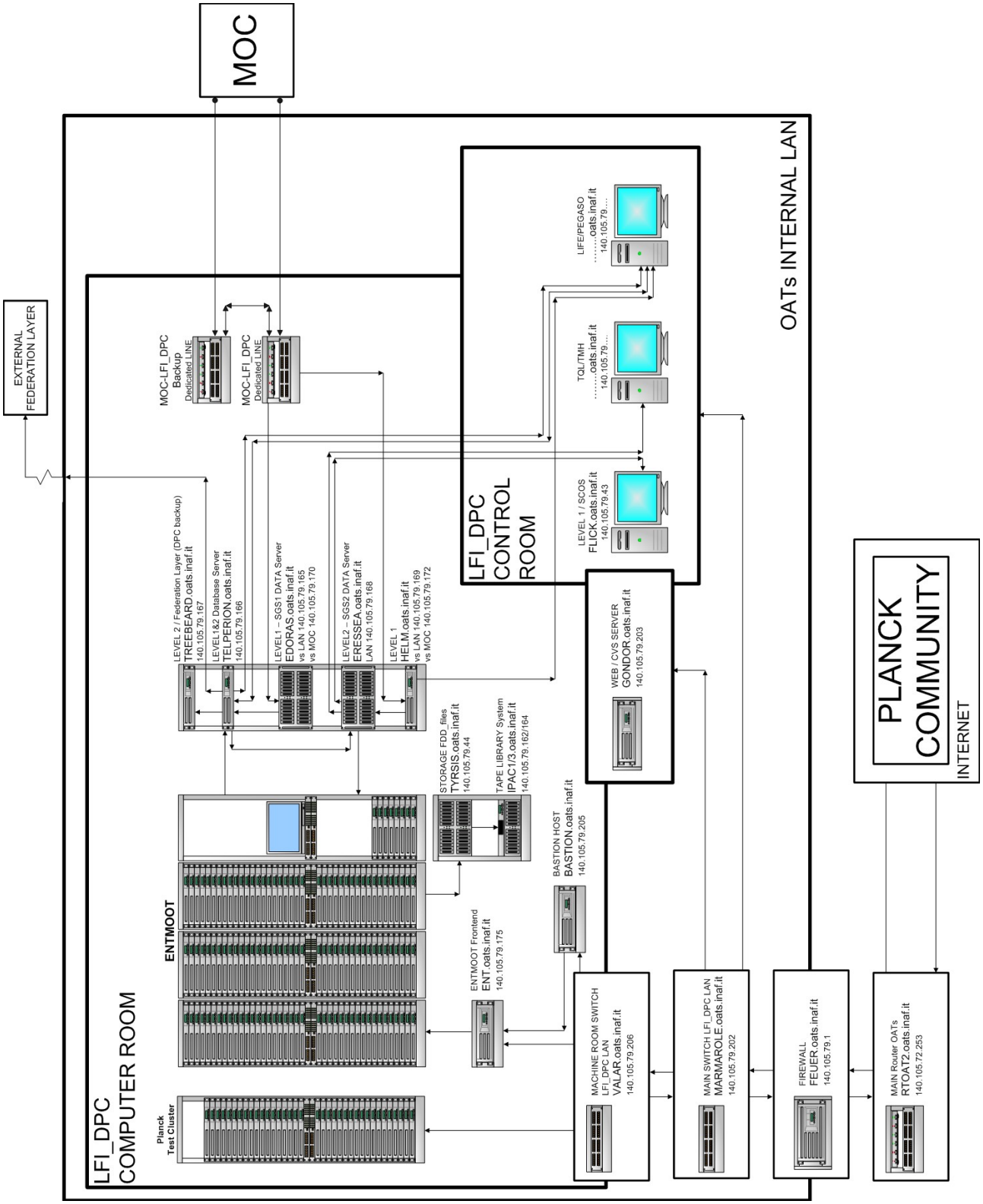

Figure 1 LFI-DPC Local Area Network Setup

Subsystem Name: 1	**RTOAT2.oats.inaf.it** 140.105.72.253	Version: 1.0	Installed:	10/2001
			Verified:	Daily
Description:	OATs MAIN ROUTER			
H/W type:	ROUTER			
Producer:	CISCO			
Model:	CISCO2601			
RIDs:				

Subsystem Name: 2	**FEUER.oats.inaf.it** 140.105.79.1/140.105.72.171	Version: 1.0	Installed:	20/09/2005
			Verified:	Daily
Description:	LFI_DPC FIREWALL			
H/W type:	Computer Pentium II 333MHz CPU, 192MB RAM, 2,5GB disk			
Producer:				
Model:				
RIDs:				
Notes:	BACKUP MACHINES AVAILABLE			

Subsystem Name: 3	**MARMAROLE.oats.inaf.it** 140.105.79.202	Version: 1.0	Installed:	10/2001
			Verified:	Daily
Description:	LFI_DPC LAN MAIN SWITCH			
H/W type:	HP SWITCH			
Producer:	HP			
Model:	HP J4093A ProCurve Switch 2424M			
RIDs:				

Subsystem Name: 4	**VALAR.oats.inaf.it** 140.105.79.206	Version: 1.0	Installed:	18/10/2004
			Verified:	Daily
Description:	LFI_DCP MACHINE ROOM SWITCH			
H/W type:	HP PROCURVE SWITCH			
Producer:	HP			
Model:	HP J4850A ProCurve Switch 5304XL			
RIDs:				

Subsystem Name: 5	**ENT.oats.inaf.it**	Version: 1.0	Installed:	SUMMER 2008
			Verified:	
Description:	ENTMOOT CLUSTER FRONTEND			
H/W type:	Computer 2x Intel Xeon Quad-Core 2 Duo 5335, 8GB RAM, 160GB disk			
Producer:	SUPERMICRO			
Model:	SUPERMICRO X7DVL-E			
RIDs:				
Notes:	The definitive FRONTEND will be installed before August 2008, since 20/06/2007 a temporary one is available, the configuration is: Computer 2x Intel Xeon 3.00GHz, 2GB RAM, 32 GB disk			

Subsystem Name: 6	**ENTMOOT** (Private Network)	Version: 1.0	Installed:	20/06/2007
			Verified:	16/06/2008
Description:	128 Node Cluster (256 CPUs)			
H/W type:	128 x Computer 2x Intel Xeon 3.00GHz, 2GB RAM, 32GB disk; Myrinet 1.2 Gbps			
Producer:	IBM			
Model:	IBM CLX Nocona			
RIDs:				
Notes:	36 Nodes being used at 16/06/2008			

Subsystem Name: 7	**PLANCK TEST CLUSTER** (Private Network)	Version: 1.0	Installed:	End of Sept. 08
			Verified:	
Description:	36 Nodes Cluster (72 CPUs)			
H/W type:	36 x Computer 2x Intel Xeon 3.00GHz, 2GB RAM, 32GB disk; Myrinet 1.2 Gbps			
Producer:	IBM			
Model:	IBM CLX			
RIDs:				

Subsystem Name:	**EDORAS.oats.inaf.it**	Version: 1.0	Installed:	05/05/2008
8	140.105.79.165/140.105.79.170		Verified:	16/06/2008
Description:	LEVEL1 - SGS1 Data Server			
H/W type:	Computer 2 x Intel Dual Core Duo E6320, 4GB RAM, 14TB disk			
Producer:	SUPERMICRO – ARECA			
Model:	SUPERMICRO PDSMA – ARECA PCI-X 16 SataII RAID			
RIDs:				
Notes:	14 TB already available, other 14TB (backup) will be bought before August 2008			

Subsystem Name:	**TELPERION.oats.inaf.it**	Version: 1.0	Installed:	27/11/2006
9	140.105.79.166		Verified:	16/06/2008
Description:	LEVEL 1 – 2 ORACLE DATABASE SERVER			
H/W type:	Computer 2x AMD Opteron DUAL CORE 2GHz CPU, 4GB RAM, 250GB disk			
Producer:	DELL			
Model:	Dell PowerEdge SC 1435			
RIDs:				

Subsystem Name:	**ERESSEA.oats.inaf.it**	Version: 1.0	Installed:	16/05/2008
10	140.105.79.168		Verified:	16/06/2008
Description:	Level 2 Storage Server (DMC server)			
H/W type:	Computer 2x Intel Dual Core Duo E6320, 4GB RAM, 14TB disk			
Producer:	SUPERMICRO – ARECA			
Model:	SUPERMICRO PDSMA – ARECA PCI-X 16 SataII RAID			
RIDs:				
Notes:	14 TB already available, other 20TB (backup) will be bought before August 2008			

Subsystem Name:	**TYRSIS.oats.inaf.it**	Version: 1.0	Installed:	14/03/2007
11	140.105.79.44		Verified:	05/06/2008
Description:	STORAGE FDD_files and FTP Server			
H/W type:	Computer 2 x Intel Xeon 2,6GHz, 2GB RAM, 120GB + 800GB + 1.5TB disk			
Producer:	SUPERMICRO – ARECA			
Model:				
RIDs:				

Subsystem Name:	**TREEBEARD.oats.inaf.it**	Version: 1.0	Installed:	14/03/2007
12	140.105.79.167		Verified:	05/06/2008
Description:	LEVEL2 Federation Layer and DMC backup			
H/W type:	Computer Intel Pentium III 1133MHz CPU, 1GB RAM, 3GB disk			
Producer:				
Model:				
RIDs:				

Subsystem Name:	**FLICK.oats.inaf.it**	Version: 1.0	Installed:	15/05/2008
13	140.105.79.43		Verified:	16/06/2008
Description:	LEVEL 1 SCOS2000			
H/W type:	Computer Intel Pentium IV 2.80 GHz, 1GB RAM, 80GB disk			
Producer:	MAXDATA			
Model:				
RIDs:				

Subsystem Name:	**GOLLUM.oats.inaf.it**	Version: 1.0	Installed:	03/12/2004
14	140.105.79.14		Verified:	16/06/2008
Description:	TQL/TMH Server			
H/W type:	Computer Intel Pentium IV 3.00 GHz, 1GB RAM, 300GB disk			
Producer:				
Model:				
RIDs:				

Subsystem Name:	**GONDOR.oats.inaf.it**	Version: 1.0	Installed:	09/06/2006
15	140.105.79.205		Verified:	13/06/2008
Description:	WEB, CVS Server			
H/W type:	Computer Intel Pentium IV 2.80 GHz, 1GB RAM, 80GB disk			
Producer:				
Model:				
RIDs:				

Subsystem Name:	**BASTION.oats.inaf.it**	Version: 1.0	Installed:	15/04/2007
16	140.105.79.205		Verified:	12/05/2008
Description:	BASTION HOST FOR remote connections			
H/W type:	Computer Intel Pentium IV 2.80 GHz, 1GB RAM, 10GB disk			
Producer:				
Model:				
RIDs:				