

LFR Source Code Analysis  
 Version 1.0  
 Solar Orbiter Mission  
 RPW INVESTIGATION  
 MEB (Main Electronic Box) Instrument  
 LFR (Low Frequency Receiver) Sub-Instrument

Iss	Rev	Date	Author	Notes
1	0	15/11/12	WRE	First Issue : analyze for Release 1
2	0	11/07/14	WRE	Updated for release 2

Prepared by:	Function:	Signature:	Date
William RECARTE, NEXEYA C&F	LFR SQA Manager for LPP		
Verified by:	Function:	Signature:	Date
Vincent LERAY, NEXEYA C&F	RPW LFR PAM for LPP		
Approved by:	Function:	Signature:	Date
Paul LEROY, LPP CNRS	RPW LFR SW Project Architect & Manager		
For application:	Function:	Signature:	Date

## Table of Contents

1	Introduction.....	4
2	Applicable and reference documents .....	4
2.1	Applicable documents .....	4
2.2	Reference documents.....	4
3	Terms, definitions and abbreviated terms .....	4
4	Methods andTools.....	6
4.1	Tool configuration .....	6
4.2	Automatic Verification Configuration.....	6
4.2.1	Metrics.....	6
4.2.2	Rules.....	7
5	Analysis results .....	10
5.1	Metrics and criteria analysis .....	10
5.1.1	Metrics.....	10
5.1.2	Criteria.....	10
5.2	Metrics analysis .....	12
5.2.1	C++ Rules.....	12
5.2.2	C Rules .....	15
6	Action plan.....	17

List of Figures

N/A

Aucune entrée de table d'illustration n'a été trouvée.

List of Table

**Aucune entrée de table d'illustration n'a été trouvée.**

## 1 Introduction

This document presents the results of analysis on the LFR Software.

This analyse consists in:

- Measurement of principal application characteristics, according to metrication defined in [AD01] and [AD02]
- Verification of software rules respect, according to coding rules defined in [AD01]

In [AD01], rules which are applicable to the LFR Software are listed. Information concerning the rule coverage concerns only these applicable rules.

## 2 Applicable and reference documents

### 2.1 Applicable documents

	Reference	Title
AD	RPW-MEB-LFR-PLN-067 V1.1	Code Rules and Metrics Analysis and Applicability to the LFR Project
AD	RPW-SYS-MEB-MGT-QAD-000405-LES V1.23	SW Quality model and Metrics

### 2.2 Reference documents

	Reference	Title
RD	ECSS-Q-ST-80 C	Space product assurance Software product assurance
RD	ECSS-E-ST-40 C	Space engineering - Software

## 3 Terms, definitions and abbreviated terms

Abbreviation	Mean
AMBA	Advanced Microcontroller Bus Architecture
DAS	DPU Application Software
DBS	DPU Boot Software
DMS	Data Management System
CEB	Central Electronic Box
CCSDS	Consultative Committee for Space Data Systems
CTR	Central Time Reference
DPU	Digital Processing Unit
DRD	Document Requirements Definition
ECSS	European Cooperation on Space Standardization
EEPROM	Electrically Erasable Programmable Read-Only Memory
EID	Experiment Interface Document
FSW	Flight Software
ICD	Interface Control Document
LSB	Low Significant Bit
MSB	Most Signification Bit
RMAP	Remote Memory Access Protocol

Abbreviation	Mean
RPW	Radio and Plasma Waves
SCET	SpaceCraft Elapsed Time
SGSE	Software Ground Support Equipments
SID	Structure IDentification
SoC	System on Chip
SRS	Software Requirements Specification
SSS	System Software Specification
TDS	Time Domain Sampler
TNR-HFR	Thermal Noise Receiver High Frequency Receiver
KP	Key Point
RB	Requirement baseline
TS	Technical Specification
DDF	Design Definition File
DJF	Design Justification File
MGT	ManaGemenT
MF	Maintenance File
PAF	Product Assurance File
SPR	Software Problem Report
NCR	Non-Conformance Report
SVS	Software Verification Specification
SVR	Software Verification Report
SPAP	Software Product Assurance Plan
SVP	Software Validation Plan
SPAMR	Software Product Assurance Milestone Report
SUITP	Software Unit an Integration Test procedure
SRF	Software Reuse File
SCF	Software Configuration File
SDP	Software Development Plan
FDIR	Failure Detection, Isolation and Recovery
SDD	Software Design Document
SRevP	Software Review Plan

## 4 Methods and Tools

### 4.1 Tool configuration

Metrics values listed into this document are obtained thanks to Logiscope V2012 tool.  
Software rules verification is done thanks to Logiscope V2012 tool.

Logiscope is configured with:

- RNC\_Scientist\_level\_C\_D\_V0.4.ref for metrication analysis
- C\_RNC\_scientist\_rules\_V0.1.rst and Cpp\_RNC\_scientist\_rules\_V0.2.rst for rules analysis

### 4.2 Automatic Verification Configuration

#### 4.2.1 Metrics

The following metrics has been verified with the following thresholds according to [AD02]:

Metric	Name	Min	Max
com_freq	Comment frequency	0.15	+oo
ct_nest	Maximum nesting level	0	4
ct_vg	Cyclomatic number (VG)	0	20
dc_calls	Number of direct calls	0	20
ic_param	Number of parameters	0	7
lc_stat	Number of statements	0	200
ap_eloc	Application effective lines of code	-oo	+oo

According to those metrics, the following criteria has been defined in [AD02]:

Criteria	Metrics	Level of Performance							
		EXCELLENT		GOOD		MEDIUM		POOR	
		Min	Max	Min	Max	Min	Max	Min	Max
Maintenability	lc_stat + ct_vg + ct_nest + dc_calls + ic_param	5	5	4	4	3	3	0	2
Autodescription	Com_frq	1	1	-	-	-	-	0	0

Note: Performance levels are directly linked to number of metrics fulfilled.

For example Maintenability criteria:

- 5 metrics fulfilled: Maintenability criteria level is Excellent
- 4 metrics fulfilled: Maintenability criteria level is Good
- 3 metrics fulfilled: Maintenability criteria level is Medium
- Less than 3 metrics fulfilled: Maintenability criteria level is Poor

## 4.2.2 Rules

Coding standard [AD01] is under configuration into Logiscope tool:

Applicability	Rules count	% implemented
C & C++	52	44.23%
C++	2	100%
C	36	63.89%
Embedded C	2	0%

Implemented rules are listed here bellow. Some of these rules might have been divided in two or more Logiscope rules in order to be completely analysed.

Rule name and version	Summary	Severity
<b>C &amp; C++ Common Rules</b>		
Don.AllocDynBord	Dynamic memory allocation is prohibited	Mandatory
Don.Declaration	All data used must be explicitly declared	Mandatory
Don.Homonymie	The use of homonyms must be avoided except in cases of overload or explicit redefinition.	Mandatory
Don.Initialisation	Variables must be initialised before being used for the first time.	Mandatory
Don.Invariant	Constants must be defined for entities whose value is invariant.	Mandatory
Don.PointeurNonAff	If the language supports the pointer concept, when a pointer is not associated with a specific object at declaration, a comment must specify the object that will be associated with it and, if the language allows, initialise it to null.	Mandatory
Don.Separee	Each piece of data must have a separate declaration.	Mandatory
Don.Utilisee	All data that is defined must be used; a datum that is no longer used must be deleted.	Mandatory
Dyn.Abort	A program must never be abruptly ended by a task or thread termination instruction (such as exit or abort).	Mandatory
Int.CheminAbsolu	Access paths must not make any hypotheses on the the current directory.	Mandatory
Org.Masquage	Data usage links should be avoided: read-and write-access operations should be used instead (information masking and data encapsulation principle), when this principle is not overly prejudicial for the language used.	Mandatory
Pr.CartStd	A standard comment box defined for the	Mandatory

Rule name and version	Summary	Severity
	project must be used to comment on the header of each module and the definition of an operation.	
Pr.Instruction	There should be no more than one instruction per line.	Mandatory
Qa.TestRetour	Function return must be systematically tested, specifically system function return.	Mandatory
Tr.BoucleSortie	A loop must feature a unique nominal exit.	Mandatory
Tr.ComparaisonStrict	Strict comparison (equality, difference) between floating numbers (real, complex) must be replaced by inequality.	Mandatory
Tr.FonctionSortie	A function must only contain one exit instruction.	Mandatory
Tr.Goto	The unconditional branching instruction (goto) must only be used in very limited and specific cases.	Mandatory
Tr.ModifCompteur	The loop counter must not be modified in loop processing.	Mandatory
Tr.ModifCondSortie	The loop exit condition must not be modified in loop processing.	Mandatory
Tr.ModifConst	The value of a constant must not be modified.	Mandatory
Tr.Parenthèses	Expressions must be systematically enclosed in parentheses.	Mandatory
Tr.Residus	No programming residue must exist as comments in the code: an instruction that is no longer used must be deleted.	Mandatory
Tr.TestEgalite	Use of the equality or difference test must be replaced by inequality where possible.	Mandatory
C++ Rules		
Don.TypeBooléen	The type bool (Boolean: true, false) must be used.	Mandatory
Tr.ModelFonction	Function models must be preferred to macros.	Mandatory
C Rules		
Don.BlocDebut	Variables must not be defined in a block outside the declarative part of the block.	Mandatory
Don.GlobDecl	It is forbidden to declare a global variable inside a subprogram.	Mandatory
Don.GlobInc	All external variables must be declared in a single file, which will be included.	Mandatory
Don.TabDyn	The dynamic dimensioning of tables must not be used.	Mandatory
Don.TypeBool	The Boolean type must be used, and must be defined if not supplied by the compiler.	Mandatory
Id.Redef	It is forbidden to redefine the keywords of	Mandatory



Rule name and version	Summary	Severity
	the language.	
Tr.Accolades	The blocks of a control structure must be enclosed in brackets "{ }", even in single-instruction blocks.	Mandatory
Tr.AffCondCall	The allocation operator "=" must not appear in a subprogram call or in a conditional expression.	Mandatory
Tr.AffectationComp	Simple allocations must replace composite allocations	Mandatory
Tr.AutoIncl	The #ifndef directive is used to avoid multiple inclusions.	Mandatory
Tr.CoherParam	The type, number and order of the effective parameters must be identical to the type, number and order of the formal parameters.	Mandatory
Tr.CondWhile	The initial value of the while loop parameter must be known before entering the loop.	Mandatory
Tr.Continue	The continue instruction is forbidden.	Mandatory
Tr.DecalSignes	The ">>" operator must not be used with signed integers.	Mandatory
Tr.FoncInterne	Internal subprograms must be declared with the static qualifier.	Mandatory
Tr.IncCode	The #include directive must not be used to include processing code.	Mandatory
Tr.IndepParam	Correlated expressions must not be placed in the call parameters.	Mandatory
Tr.InstBordCommun	Subprograms that modify common data must not be used in the same subprogram instruction.	Mandatory
Tr.MultAff	Multiple allocation is forbidden.	Mandatory
Tr.OperCond	The alternatives operator is forbidden.	Mandatory
Tr.ParAnonym	Parameters must be named in the prototype declaration of a subprogram.	Mandatory
Tr.ParenArg	Macros must be parenthesised.	Mandatory
Tr.PostFix	The postfix use of "++" and "--" must be restricted to simple cases. Prefix use is forbidden.	Mandatory
Tr.Pragma	Avoid using the #pragma directive.	Mandatory
Tr.RefComplexite	The level of complexity of a reference must be limited.	Mandatory
Tr.TailleMacro	Macros must have no more than five instructions.	Mandatory
Tr.TypeFonc	The type of a subprogram must be explicitly defined.	Mandatory
Tr.VarArgs	It is forbidden to define a subprogram or a macrofunction with a variable number of	Mandatory

Rule name and version	Summary	Severity
	parameters.	
C Embedded Rules		
	No implemented rule yet	

## 5 Analysis results

### 5.1 Metrics and criteria analysis

Only summary and analysis of metrics out of the thresholds and violated rules is given in this document.

Full analysis report is provided in HTML format in appendix.

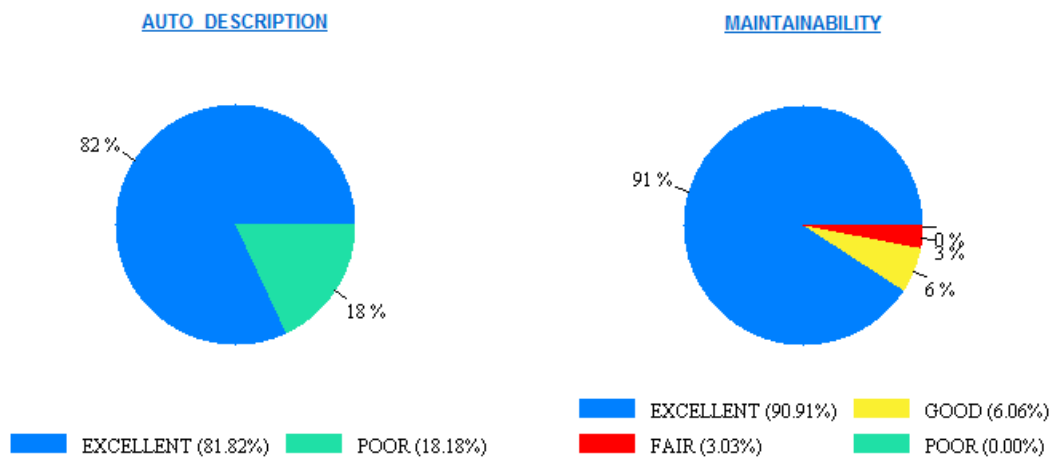
#### 5.1.1 Metrics

Mnemonic	Metric Name	Min	Max	Out
com_freq	Comment frequency	0.15	+oo	18.18%
ct_nest	Maximum nesting level	0	4	5.05%
ct_vg	Cyclomatic number (VG)	0	20	4.04%
dc_calls	Number of direct calls	0	20	0.00%
ic_param	Number of parameters	0	7	0.00%
lc_stat	Number of statements	0	200	3.03%
ap_eloc	Application effective lines of code	-oo	+oo	6624

All metrics out of the thresholds are less than 5% deviation which was agreed as acceptable for this version.

**Action AI-1:** Metrics results out of 5% will be corrected before CDR R2.

#### 5.1.2 Criteria



Criteria are directly linked to metrics analysis. Here after are listed all functions per criteria with a Fair and Poor criteria.

**Note:** Auto-description and Maintainability criteria will be increased after AI-41 taken into account.

#### 5.1.2.1 Auto description: POOR

Functions & Methods name
SM_average (float*, float*, ring_node_sm**, unsignedint, unsignedint)
SM_average_f2 (float*, ring_node_sm*, unsignedint)
lecture_file_sm (const char*)
init_k_coefficients_f0 (void)
get_v_e1_e2_f3 (unsignedchar*)
ASM_generic_init_ring (ring_node_asm*, unsignedchar)
SM_reset_current_ring_nodes (void)
set_time (unsignedchar*, unsignedchar*)
close_matrix_actions (unsignedint*, unsignedint, rtems_id, ring_node_sm*, ring_node_sm*, unsignedlonglongint)
check_update_info_hk_tds_mode (unsignedchar)
check_update_info_hk_thr_mode (unsignedchar)
init_header_continuous_wf_table (unsignedint, Header_TM_LFR_SCIENCE_CWF_t*)
init_header_continuous_cwf3_light_table (Header_TM_LFR_SCIENCE_CWF_t*)
compute_acquisition_time_old (unsignedint, unsignedint, unsignedint, unsignedchar, unsignedchar*)
compute_acquisition_time (unsignedint, unsignedint, unsignedint, unsignedchar, unsignedchar*)
set_wfp_delta_f0_f0_2 (void)
set_wfp_delta_f1 (void)
set_wfp_delta_f2 ()

#### 5.1.2.2 Maintainability: FAIR

Functions & Methods name
BP1_set (float*, float*, uint8_t, uint8_t*)
BP2_set (float*, uint8_t, uint8_t*)
spiq_task (rtems_task_argument)

## 5.2 Metrics analysis

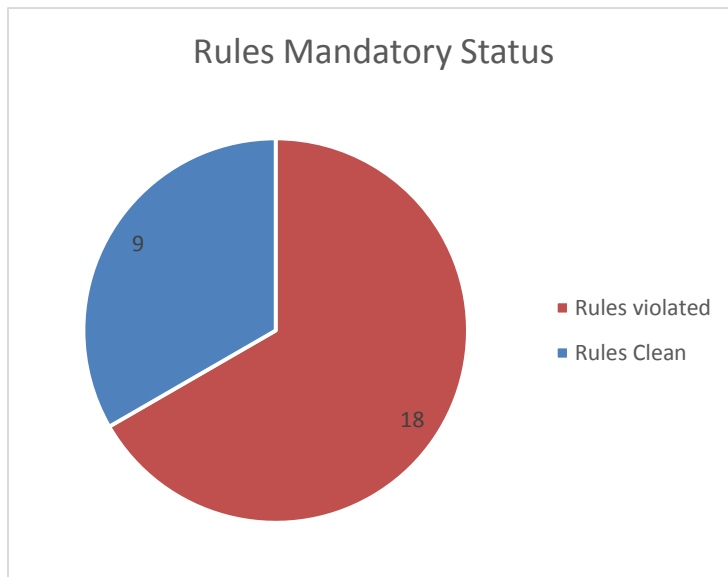
**Action AI-2:** Correct positive deviation to rules before CDR R2.

**Action AI-3:** Analyse unstatus rules before CDR R2.

### 5.2.1 C++ Rules

The following rules are marked violated by Logiscope:

#### 5.2.1.1 Summary



#### 5.2.1.2 Don.Homonymie

Rule is violated 49 times.

basic\_parameters.h wrong positives due to use on “printf”.

ccsds\_types.h: wrong positive, use of spare1\_pusVersion\_spare2 both structs :  
Packet\_TM\_LFR\_TC\_EXE\_SUCCESS\_t and  
Packet\_TM\_LFR\_TC\_EXE\_INCONSISTENT\_t detected as an error by Logiscope

fsw\_params.h: Positive error, same string is used in upper case for #define and in lower case for variables name when there are used in “tc\_load\_dump\_parameters.c”.

fsw\_processing.h: Positive error, same string is used for table name (Task\_id) and variable name (task\_id)

fsw\_processing\_globals.c: wrong positives

fsw\_spacewire.h: wrong positive (name used in function parameter)

lfr\_cpu\_usage\_report.c: wrong positive (due to unknown type in Logiscope)

main.c: wrong positive

tc\_acceptance.c: Wrong positives (name used in function parameter)

wf\_handler.c: Wrong positives (name used in function parameter)

#### **5.2.1.3 Don\_Initialisation**

Rule is violated 152 times.

Some wrong positives due to :

- Initialisation by calling a function with parameter passed by reference (in / out parameter)
- Initialisation by return of a called function.

All other cases shall be analyzed.

#### **5.2.1.4 Don.Invariant**

Rule is violated 2344 times.

Rule raise an error every time a number is used :

- Case of comparison to an integer value
- Case of table index use
- Case of variable affectation

Those cases are considered as wrong positives.

#### **5.2.1.5 Don\_PointeurNonAff**

Rule is violated 27 times.

To be corrected, all mentioned pointeur are not initialized to NULL.

#### **5.2.1.6 Don\_Separee**

Rule is violated 2 times.

To be corrected

#### **5.2.1.7 Don\_TypeBooleen**

Rule is violated 4 times.

To be corrected : “if ( information == true)” instead of if “( information )”

#### **5.2.1.8 Don.Utilisee**

Rule is violated 596 times.

It seems that at each time a variable is defined inside a struct it activates this rule. Maybe we are in presence of false positives. Nevertheless, it should be assured that each variable is indeed used in the program.

A fast check should be done to each error alert.

#### **5.2.1.9 Int.CheminAbsolu**

Rule is violated 4 times.

To be corrected.

#### **5.2.1.10Pr\_Instruction**

Rule is violated 7 times.

Due to use of debug instruction:  
if (XXX) PRINTF("YYY")

To be corrected.

#### **5.2.1.11Tr\_BoucleSortie**

Rule is violated 3 times.

Due to use of “continue” instruction

To be corrected.

#### **5.2.1.12Tr\_ComparaisonStrict**

Rule is violated 14 times.

To be corrected.

#### **5.2.1.13Tr\_FonctionSortie**

Rule is violated 10 times.

To be corrected. (more than one return or no return in a non void function)

#### **5.2.1.14Tr.ModelFonction**

Rule is violated 18 times.

Due to use of debug macro to redefine printf.  
To be corrected.

#### **5.2.1.15 Tr\_ModifCompteur**

Rule is violated 2 times.

Wrong positive

#### **5.2.1.16Tr\_Parenthèses**

Rule is violated 2 times.

Rule to be modified with partpar parameter, for explanation, see Das issue : [RPWDPU-851](#)

#### **5.2.1.17Tr.Residus**

Rule is violated 12 times.

Code should be deleted instead of commented. To delete.

#### **5.2.1.18Tr\_TestEgalite**

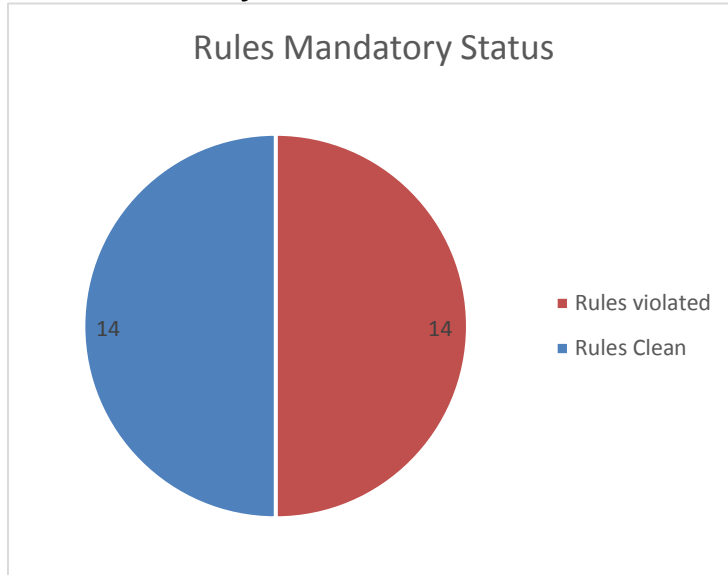
Rule is violated 14 times.

All violation enter in context of exception mentioned into rule description.

## 5.2.2 C Rules

The following rules are marked violated by Logiscope:

### 5.2.2.1 Summary



#### 5.2.2.2 *Don\_GlobDecl*

Rule is violated 2 times.

To be corrected.

#### 5.2.2.3 *Don.TypBool*

Rule is violated 1 time.

Wrong positive.

#### 5.2.2.4 *Qa.TestRetour*

Rule is violated 4 times.

Four “set” functions, and only one takes a value as input. Do these functions return anything (true or false for instance)? To check.

#### 5.2.2.5 *Id\_Redef\_partial2*

Rule is violated 2 times.

Wrong positives.

#### 5.2.2.6 *Qa\_TestRetour*

Rule is violated 7 times.

To be corrected.

#### 5.2.2.7 *Tr\_Accolades*

Rule is violated 1 time.

Wrong positive.

#### **5.2.2.8 *Tr\_CoherParam\_partial1***

Rule is violated 10 times.

To be analysed.

#### **5.2.2.9 *Tr\_CoherParam\_partial2***

Rule is violated 10 times.

To be analysed.

#### **5.2.2.10 *Tr\_DecalSignes***

Rule is violated 12 times.

Wrong positives (cast not seen by Logiscope tool)

#### **5.2.2.11 *Tr.FoncInterne***

Rule is violated 144 times.

To be analysed.

#### **5.2.2.12 *Tr.IncCode***

Rule is violated 1 times

Probably a template, to be analysed.

#### **5.2.2.13 *Tr\_OperCond\_***

Rule is violated 2 times.

To be corrected.

#### **5.2.2.14 *Tr\_ParenArg\_p1***

Rule is violated 15 times.

Due to use of debug macro to redefine printf.

To be corrected.

#### **5.2.2.15 *Tr\_ParenArg\_p2***

Rule is violated 18 times.

Due to use of debug macro to redefine printf.

To be corrected.

#### **5.2.2.16 *Tr.TypeFonc***

Rule is violated 21 times.

To be analysed.



## 6 Action plan

Action ID	Action	Due date
AI-1	Metrics results out of 5% will be corrected	CDR R2
AI-2	Correct positive deviation to rules	CDR R2
AI-3	Analyse unstatus rules	CDR R2