HERSCHEL/PACS

SPU HIGH LEVEL SOFTWARE

Specification Document

Document Ref.: PACS-TW-GS-001

Issue: 4.8

Prepared by: **Ahmed Nabil BELBACHIR** Date: **18. Feb 2009**

Roland OTTENSAMER

Christian REIMERS

Checked and Approved by: **Franz KERSCHBAUM**

Checked and Approved by: **Helmut FEUCHTGRUBER**

Distribution Record

Issue / Revision	Draft 0	Draft 1	Draft 2	1.0	1.1	1.2	1.3
Distribution Date	16.07.2000	29.08.2000	16.03.2001	22.05.2001	02.09.2001	15.11.2001	29.01.2002
Issue / Revision	1.4	2.0	2.1	2.2	3.0	4.0	4.1
Distribution Date	08.04.2002	06.08.2002	12.09.2002	17.02.2003	10.03.2003	23.09.2003	13.10.2003
Issue / Revision	4.2	4.3	4.4	4.5	4.6	4.7	4.8
Distribution Date	27.02.2004	30.03.2004	13.04.2006	04.09.2006	05.11.2007	01.07.2008	18.02.2009

INTERNAL			**EXTERNAL**		
Department	**Name**	**Qty**	**Company**	**Name**	**Qty**
UVIE/ASTRO	F. Kerschbaum	1	MPE	H. Feuchtgruber, O. Bauer, E. Wieprecht, A. Contursi, T. Müller, G. Wildgruber	1
			CSL	J.M. Gillis, A. Mazy	1
			IAC	J.M. Herreros, P. Gomez	1
			IFSI	R. Orfei, S. Pezzuto	1
			MPE	PACS Project Office (pacs@mpe.mpg.de) PACS Warm Electronic (pacs-we@ster.kuleuven.ac.be)	1
			Electronic Archives at Leuven http://pacs.ster.kuleuven.ac.be		1

Document Change Records

Document Title **HERSCHEL – PACS** **SPU High Level Software Document**		
Issue	**Date**	**Reason for Change**
Draft 0	15/07/2000	Initial Issue
Draft 1	22/08/2000	Comments from MPE
Draft 2	16/03/2001	Comments from Consortia and OBS#4, document structure update
1.0	16/05/2001	Comments from the SPU HLSW Meeting #1
1.1	02/09/2001	Comments from OBS#5
1.2	28/10/2001	Inclusion of the HLSW Design
1.3	29/01/2002	Update taking into account new version of the documents
1.4	08/04/2002	Include the SW Tables structure
2.0	05/08/2002	Comments from IBDR, reorganized
2.1	11/09/2002	Comments from SPU HLSW Version 1 AVM Delivery
2.2	17/02/2003	• New Perform Activity (RCS) added, description of expected TM Rates for all Compression Modes in Phot./Spec. added, CEH size changed to 28 Bytes, detailed Label description inserted, parameters RCXNB and RCX in DET_CST_SPEC/DET_CST_PHOT tables deleted. • Triple and Quadruple Compression Modes in photometry and spectroscopy are removed because Default and Double Compression Mode fulfil the TM requirements. Further integration may be performed on ground if required • Lossless Coding Only Compression Mode in photometry and spectroscopy is removed. This mode is already covered by The Lossless Compression Mode. • SPU Test Compression Mode in photometry and spectroscopy is removed. This mode is already covered by The Default Compression Mode + Raw Channel, and by the SPU Internal Test Mode (where data are generated inside the SPU).
3.0	10/03/2003	Updated for SPU HLSW Version 3.0

4.0	23/09/2003	• Updated for SPU HLSW Version 6.0 • Include two new DPU commands: "Copy data from RAM to EEPROM" and "Connect to DEC/MEC". • Update the logical model (Watch Process and Command Interpretation and Acknowledgment) by including the two new commands • Include a description of SPU HLSW SMCS handling in Section 5.1. • Include a list of HLSW HK parameters and update the HK rate in Section 5.5 • Compression SW Scheme updated (Figure 16) • Raw Channel data compression included (Appendix A.1.3 and Section 5.9.1) • Updated description for ramp fitting in Spectroscopy (Section 5.9.4 and Appendix B.2.2.7) where several slopes can derived out of a ramp. • Update the Section 6 "Feasibility and Resource Estimates", and delete the decompression SW description from this Section • Update Section 7 and 8 "Traceability Matrices" • Include a CRISA document as reference document "RD015" • Update the tables for the Write command in Section 5.7.11
4.1	13/10/2003	• Include software specifications for lossless compression ode (Section3.4.3.1.4) • Update the traceability matrices according to the added scpecifications above (Section78) • Update Section 5.9.6 according to the specifications above
4.2	27/02/2004	• Include a parameter for the Half Compression Mode in Photometry (20Hz readout rate) • Modify the DIFS field in Compressed entity Header by REAL (reduction algorithm) used on-board. • Include two parameters for the Write command in spectroscopy (Reduction algorithm to use, the use of glitch or not) • Update the appendixes for the used algorithms in HLSW version 8. • Include the description of new data sorting algorithm in Appendix (Reorder)

		• Include in Appendix the description of subramp fitting • Update the TM rate calculated for Buffer transmission mode • after increasing the transmission rate (SPU-DPU) in photometry and after including the TM packet header
4.3	26/03/2004	• Include SW requirements SPU-HSR-FN0202 and SPU-HSR-FN0203 • Update the HK Table parameters Section 5.5. • Update Fill_InputBuffer Section 5.6 including the functions used for mrmory scrubbing. • Updated Section 5.8.5 for Buffer Transmission Mode in Photometry • Noise resampling algorithm is implemented in the peak-up module. Update the the Peak-up description Section 5.13. • Update the traceability matices in Section7 and 8 include the new software requirements
4.4	13/04/2006	• Include SW requirements (for FDIR) SPU-HSR-SA0010, SPU-HSR-SA0011 and SPU-HSR-SA0012 • Update the traceability matix in 8 include the new software requirements • Include Section 9 Software Requirements vs. Software Verification & Validation Traceability Matrix • Implementation of the reworked CASE Tool diagrams in Section 2.2. • Update llc.c component description (Section 5.9.7) • Update Section A.1 of currently implemented algorithms for the lossless compression • Update Section A.2 of library of available algorithms for the lossless compression • Update Section B.2 of library of available algorithms for the fitting algorithms
4.5	01/09/2006	• Update and include SW requirements according to CAPTEC code review comments (see 588-200b_SPU_SSD_Comments.xls) • Update Section 9 (Software Requirements vs. Software Verification & Validation Traceability Matrix) • Update Sections according to CAPTEC code review comments (see 588-200b_SPU_SSD_Comments.xls)
4.6	05/11/2007	• Adaption of requirements to HLSW version 13.8 • Apply new naming conventions from the “cleaned version” (e.g. bol1_2 → bol_ex, …) • Add new algorithms in Section A.1: RZIP2 • Add new algorithms in Section A.2: RZIP2, Data Resorting

		(Reorder), FM Arithmetic Compression and VBWL_short • Add new Section B.2.4: Averaging and Rounding in Photometry
4.7	01/07/2008	• Added Compressed Sensing mode short description (table 4) • Added Section “FM Averaging” in appendix B
4.8	18/02/2009	• Added appendix B.3: Decimation

Contents

1 Introduction ... 16
1.1 Purpose of the document ... 16
1.2 Definitions, Acronyms and Abbreviations ... 16
1.3 References ... 20
1.3.1 Applicable Documents ... 20
1.3.2 Reference Documents ... 20
1.4 Overview of the document ... 21
2 Model Description ... 22
2.1 General ... 22
2.1.1 The Signal Processing Unit (SPU) ... 23
2.1.2 The SPU High Level Software (HLSW) ... 25
2.2 Logical Model ... 26
3 Specific Requirements ... 35
3.1 Functional Requirements ... 35
3.1.1 The Watch Process ... 35
3.1.2 Application Software ... 39
3.1.3 HK Module ... 40
3.2 Performance Requirements ... 41
3.3 Interface Requirements ... 41
3.3.1 DPU Commands ... 41
3.3.1.1 LOAD Command ... 41
3.3.1.2 DUMP Command ... 42
3.3.1.3 CHECK Command ... 42
3.3.1.4 RESET Command ... 42
3.3.1.5 RAW_CHAN_TRAN_MODE Command ... 43
3.3.1.6 STOP_REDUCT_COMPR Command ... 43
3.3.1.7 START_REDUCT_COMPR Command ... 43
3.3.1.8 ACT_TEST_PHOT Command ... 43
3.3.1.9 ACT_TEST_SPEC ... 43
3.3.1.10 ACT_BOL_BGND_CANCEL Command ... 44
3.3.1.11 Write Commands ... 44
3.3.1.12 CP_DATA_RAM_EEPROM Command ... 44
3.3.1.13 CONNECT_DMC Command ... 44

3.3.2 The Compressed Entity ... 45
3.3.3 DEC/MEC to SPU Interface ... 45
3.4 Operational Requirements ... 46
3.4.1 DEC/MEC Header Usage ... 46
3.4.1.1 Spectroscopy ... 46
3.4.1.2 Photometry ... 51
3.4.2 Instrument Configuration ... 55
3.4.2.1 Instrument Operating Modes ... 55
3.4.2.2 Instrument Observing Modes ... 55
3.4.3 SPU HLSW Compression Modes ... 56
3.4.3.1 Compressed Data Transmission Mode ... 56
3.4.3.1.1 Default Compression Mode ... 57
3.4.3.1.2 Double Compression Mode ... 57
3.4.3.1.3 Half-Compression Mode (Photometry only) ... 57
3.4.3.1.4 Lossless Compression Mode ... 57
3.4.3.2 SPU Test Mode ... 58
3.4.3.3 Transparent Mode ... 58
3.4.3.4 Buffer Transmission Mode ... 58
3.4.3.5 Compression Mode for 4s Reset (Spectroscopy only) ... 58
3.5 Resource Requirements ... 59
3.6 Verification Requirements ... 59
3.7 Acceptance Testing Requirements ... 60
3.8 Documentation Requirements ... 60
3.9 Portability Requirements ... 60
3.10 Quality Requirements ... 61
3.11 Reliability Requirements ... 61
3.12 Maintainability Requirements ... 62
3.13 Safety Requirements ... 63
4 System Design ... 64
4.1 Design Method ... 64
4.2 Decomposition Description ... 64
4.2.1 The Watch Process and Command Interpretation and Acknowledgement ... 64
4.2.2 The Application SW ... 66
4.2.2.1 Supervisor ... 66
4.2.2.2 Compression Software ... 67
4.2.2.2.1 HLSW Compression Modes ... 68
4.2.2.2.2 Summary of Compression Modes ... 69

4.2.3 The HK Module ... 71
5 Component and Functional Description ... 72
5.1 Main Task ... 73
5.2 Watch Process ... 77
5.3 Supervisor ... 80
5.4 Data Transmission ... 89
5.5 Housekeeping ... 92
5.6 Fill_InputBuffer ... 94
5.7 Command Interpretation and Acknowledgement ... 96
5.7.1 LOAD Command ... 96
5.7.2 DUMP Command ... 98
5.7.3 CHECK Command ... 100
5.7.4 Perform Activity Command ... 102
5.7.5 CP_DATA_RAM_EEPROM Command ... 105
5.7.6 RESET Command ... 107
5.7.7 RAW_CHAN_TRAN_MODE Command ... 109
5.7.8 START/STOP_REDUCT_COMPR Command ... 111
5.7.9 ACT_TEST_PHOT/SPEC Command ... 113
5.7.10 CONNECT_DMC Command ... 115
5.7.11 Write Command ... 117
5.8 Compression Software ... 120
5.8.1 Compression Software Main Task ... 120
5.8.2 Photometry Modes 0, 1 and 2 ... 123
5.8.3 Photometry Mode 3 ... 125
5.8.4 Photometry Mode 4 ... 127
5.8.5 Photometry Buffer Transmission Mode ... 129
5.8.6 Spectroscopy Modes 1 and 2 ... 132
5.8.7 Spectroscopy Mode 3 ... 134
5.8.8 Spectroscopy Mode 4 ... 136
5.8.9 Spectroscopy Buffer Transmission Mode ... 137
5.9 Compression SW Modules ... 139
5.9.1 Detector Selection ... 139
5.9.2 Preprocessing ... 141

5.9.3 Robust Averaging ... 143
5.9.4 Ramp Fitting ... 145
5.9.5 Integration (obsolete) ... 148
5.9.6 Temporal and Spatial Redundancy Reduction ... 150
5.9.7 Lossless Coding ... 153
5.9.8 Header Compression ... 155
5.9.9 Fill_OutputBuffer ... 157
5.10 Supervisor for Test Mode ... 160
5.11 Photometry Data Generator ... 165
5.12 Spectroscopy Data Generator ... 167
6 Feasibility and Resource Estimates ... 169
6.1 Memory Allocation ... 169
6.2 Processing Power ... 170
6.3 Application SW Estimates ... 170
6.3.1 Compression Ratio ... 170
6.3.2 Entropy Estimates for Redundancy Reduction ... 171
6.4 Data Rates Estimate ... 172
6.4.1 Photometry ... 173
6.4.2 Spectroscopy ... 175
7 User Requirements vs. Software Traceability Matrix ... 177
8 Software Requirements vs. Components Traceability Matrix ... 179
9 Software Requirements vs. Software Verification & Validation Traceability Matrix ... 185
Appendices ... 191
A.1 Currently implemented Algorithms ... 193
A.1.1 Lossless DEC/MEC Header Compression ... 193
A.1.2 Lossless Science Data Compression ... 193
A.1.3 Raw Channel Data Compression ... 194
A.2. Library of LLC Algorithms ... 195
A.2.1 List of Available Algorithms ... 195
A.2.2 Dynamic and Static Temporal Redundancy Reduction (D_S_TRR) ... 195
A.2.3 Simple Zero Repetition Suppression (ZRS) ... 196
A.2.4 RZIP/RZIP2 ... 196
A.2.5 Arithmetic Coding (ARC) ... 198

A.2.6 Data Resorting (Reorder) 199
A.2.7 PACS Codec (new_srt) 200
A.2.8 FM Arithmetic Compression 200
A.2.9 VBWL 201
B.1 Currently implemented Algorithms 203
B.1.1 Robust Averaging in Photometry 203
B.1.2 Ramp-Fitting in Spectroscopy 203
B.2 Library of Fitting Algorithms 204
B.2.1 List of Available Algorithms 204
B.2.2 Robust Averaging in Photometry 205
B.2.2.1 Glitch Detection and Rejection 205
B.2.2.1.1 Sample Deviation Detection Method 205
B.2.2.2 Averaging 206
B.2.2.2.1 FM Averaging 206
B.2.2.2.2 Mean Value Calculation 206
B.2.3 Ramp-Fitting in Spectroscopy 207
B.2.3.1 Glitch Detection and Rejection 207
B.2.3.1.1 Crossed Slope Deviation Detection 207
B.2.3.2 Ramp-Fitting 208
B.2.3.2.1 Least Squares Fitting 208
B.2.3.2.4 Two-Sample-Fit 211
B.2.3.2.5 Slope Deviation Detection Method 212
B.2.3.2.6 Two-Samples Difference Method 215
B.2.3.2.7 Subramp Fitting 216
B.2.4 Averaging and Rounding in Photometry 217
B.3 Decimation 2037
B.3.1 Decimation in Photometry 20417
B.3.2 Decimation in Spectroscopy 20417

Index of Figures

Figure 1. PACS Instrument Block Diagram...22
Figure 2. Communication Links between SPU and other Subsystems...24
Figure 3. SPU HLSW Architecture...26
Figure 4. WP_and_CIA Module...27
Figure 5. Watch Process Module...28
Figure 6. ASW Module...29
Figure 7. CSW Module...30
Figure 8. Components of Spectroscopy Mode 1 (Spec1) Module...31
Figure 10. Components of Spectroscopy Mode 3 (Spec3) Module...32
Figure 11. Components of Spectroscopy Mode 4 (Spec4) and Photometry Mode 4 (Bol4) Module...32
Figure 12. Buffer Transmission Mode in Spectroscopy (BUF_Spec) and Photometry (BUF_Bol)...33
Figure 13. Components of Photometry Mode (Bol_ex) Module...33
Figure 15. Components of Photometry Mode 3 (Bol3) Module...34
Figure 18. Flowchart of the Supervisor Algorithm...67
Figure 19. Compression SW Scheme....70
Figure 20. Flowchart of the Main Taks....75
Figure 21. Flowchart of the Watch Process Task...78
Figure 22. Detector array for the Spectroscopy....80
Figure 23. Configuration of the (a) SWL-Detector Arrays and of the (b) LWL-Detector Arrays....81
Figure 24. Resulting DMC packet structure for Spectroscopy....81
Figure 25. Resulting DMC packet structure for (a) SWL-Photometry and for (b) LWL-Photometry....82
Figure 26. Flowchart of the Supervisor Task (Part 1)...85
Figure 27. Flowchart of the Supervisor Task (Part 2)...85
Figure 28. Flowchart of the Supervisor Task (Part 3)...85
Figure 29. Flowchart of the Data Transmission Task....90
Figure 30. Flowchart of the Housekeeping Task...93
Figure 31. Flowchart of the Fill_InputBuffer Procedure...95
Figure 32. Flowchart of the Procedure for Load Command Handling...97
Figure 33. Flowchart of the Procedure for Dump Command Handling...99
Figure 34. Flowchart of the Procedure for Check Command Handling...101
Figure 35. Flowchart of the Procedure for the Perform Activity Command Handling...103

Figure 36. Flowchart of the Procedure for Copy Data from RAM to EEPROM Command Handling..................... 106
Figure 37. Flowchart of the Procedure for Warm Reset Command Handling 108
Figure 38. Flowchart of the Procedure for Raw Channel Selection Command Handling 110
Figure 39. Flowchart of the Procedure for the Start and Stop Commands Handling 112
Figure 40. Flowchart of theProcedure for SPU Test Command Handling.. 114
Figure 41. Flowchart of the Procedure for Connect to DEC/MEC Command Handling.. 116
Figure 42. Flowchart of the Procedure for Write Command Handling.. 119
Figure 43. Flowchart of the Compression Software Main Task (Part 1).. 121
Figure 44. Flowchart of the Compression Software Main Task (Part 2).. 122
Figure 45. Flowchart of the Bol_ex Procedure 124
Figure 46. Flowchart of the Bol3 Procedure.. 126
Figure 47. Flowchart of the Bol4 Procedure.. 128
Figure 48. Flowchart of the BUF_Bol Procedure 130
Figure 49. Flowchart of the Spec1_2 Procedure.. 133
Figure 50. Flowchart of the Spec3 Procedure.. 135
Figure 51. Flowchart of the Spec4 Procedure.. 136
Figure 52. Flowchart of the BUF_Spec Procedure 138
Figure 53. Flowchart of the Detector Selection Procedure.. 140
Figure 54. Flowchart of the Preprocessing Procedure.. 142
Figure 55. Flowchart of the Robust Averaging Procedure.. 144
Figure 56. Flowchart of the Ramp-fitting Procedure 146
Figure 57. Flowchart of the Integration Procedure.. 149
Figure 58. Flowchart of the Temporal and Spatial Redundancy Reduction Procedure for Spectroscopy............... 151
Figure 59. Flowchart of the Temporal and Spatial Redundancy Reduction Procedure for Photometry.................. 152
Figure 60. Flowchart of the Lossless Compression Procedure 154
Figure 61. Flowchart of the DEC/MEC Header Compression Procedure 156
Figure 62. Splitting of the Compressed Entity and the Resulting Science Data Blocks... 157
Figure 63. Flowchart of the Fill_OutputBuffer Procedure.. 158
Figure 64. Flowchart of the Supervisor for Test Mode Task (Part 1).. 161
Figure 65. Flowchart of the Supervisor for Test Mode Task (Part 2).. 161
Figure 66. Flowchart of the Supervisor for Test Mode Task (Part 3).. 161
Figure 67. Flowchart of the Supervisor for Test Mode Task (Part 4).. 164
Figure 68. Flowchart of the Photometry Data Generator Task.. 166

Figure 69. Flowchart of the Spectroscopy Data Generator Task *168*
Figure 70. SPU HLSW Memory Map. *169*
Figure 71. Redundancy Reduction. *171*
Figure 72. Scheme of the PACS Codec algorithm. *200*
Figure 73. Example showing the sample deviation detection method *205*
Figure 77. Example showing the crossed slope deviation detection *207*
Figure 78. Least Square Fitting (1) *208*
Figure 79. Least Square Fitting (2) *208*
Figure 83. Example showing the ramp-fitting method (fit and residuals) *211*
Figure 84 Example showing the slope deviation detection method *212*
Figure 85. Photoconductor Readouts from 1 Channel with Additive Noise *213*
Figure 86. Successive Slopes in the Ramp *213*
Figure 87. Successive Slope Deviations in the Ramp *214*
Figure 88. Example showing the Two-Samples Difference Method *215*
Figure 89. Illustration of Sub-ramp Fitting *216*

Index of Tables

Table 1. PACS Instrument Operating Modes *55*
Table 2. PACS Instrument Observing Modes *56*
Table 3, Acceptance Test *60*
Table 4. Photometric and Spectroscopic compression modes *69*
Table 5. Software Files of the SPU HLSW *72*
Table 7. List of Compression Parameters for Photometry and Spectroscopy *83*
Table 12. Memory Allocation of the SPU HLSW. *169*
Table 13. Number of Bits Allocated to the Noise. *172*
Table 14. Expected Data Rates for the Compression Modes in Photometry *172*
Table 15. Expected Data Rates for the Compression Modes in Spectroscopy *173*
Table 16. Nominal Data Rates for Photometry. *174*
Table 17. Nominal Data Rates for Spectroscopy. *175*
Table 18. List of available LLC algorithms. *195*
Table 19. Keys for RZIP Combinationss *197*
Table 20. The Basic Arithmetic Coding Process *198*
Table 21. List of available fitting algorithms. *204*

1 Introduction

1.1 Purpose of the document

The purpose of this document is to describe the software specifications of the SPU HLSW developed by UVIE/TUVIE according to the ESA Software Engineering Standards described in RD001, which includes the Software Requirement Definition Phase (SR), Architectural Design Phase (AD) and the Software Design Phase (SD). These specifications include the SW concept, design, requirements and library of data reduction algorithms.

The SPU HLSW consists mainly of the ASW (Reduction/Compression), the watch process (it acts upon DPU commands) and communication protocols with other sub-units. The rest of the SPU development, which consists of HW-LLSW, is part of the IAC responsibility and out of the scope of this document.

1.2 Definitions, Acronyms and Abbreviations

Acronyms and Abbreviations	Description
ADSP	Analog device DSP
AID	Activity ID
AOCS	Attitude and Orbit Control Subsystem
ASW	Application SoftWare
AVM	AVionics Model
BOL	BOLometers
CDMS	Command and Data Management System
CPU	Central Processing Unit
CQM	Cryogenic Qualification Model
CRE	Cold Readout Electronics
CSL	Centre Spatial de Liège
CVS	Concurrent Versions System
DDR	Detailed Design Review
DEC/MEC	Detector Controller/Mechanisms Controller
DMC	DEC/MEC
DMCH	DEC/MEC Header
DPU	Digital Processing Unit
DRCU	Detector Readout and Control Unit
DSP	Digital Signal Processor
EDAC	Error Detection And Correction
EEPROM	Electrically Erasable Programmable Read Only Memory
FCU	FPU Control Unit
FIRST	Far Infrared and Sub-millimetre Telescope
FPU	Focal Plane Unit
FS	Flight Spare (Model)
HIFI	Heterodyne Instrument for HERSCHEL (HiFi instrument)
HK	House-Keeping
HLSW	High Level SW

HS	High Speed
HSO	Herschel Space Observatory
HW	HardWare
I/F	Interface
IAC	Instituto de Astrofísica de Canarias
ID	Identification
IFSI	Istituto di Fisica dello Spazio Interplanetario
LLSW	Low Level SW
LSB	Least Significant Byte
LWL	Long WaveLength
MFCU	Mechanisms and Focal plane electronics Control Unit
MSB	Most Significant Byte
OBCP	On-Board Control Procedure
OBS	On-Board Software
PACS	Photo-detector Array Camera and Spectrometer
PFM	Proto-Flight Model
PHC	PhotoConductors
PMA	Program Memory Address
PMD	Program Memory Data
PRIP	Pattern Recognition and Image Processing
PROM	Programmable Read Only Memory
RAM	Random Access Memory
ROM	Read Only Memory
SAU	Smallest Addressable Unit
SCR	Software Change Request
SDE	Software Development Environment
SEU	Single Error Upsets
SID	Structure ID
SPIRE	Spectral and Photometric Imaging Receiver
SPR	Software Problem Report
SPU	Signal Processing Unit
SW	SoftWare
SWL	Short WaveLength
TBC	To Be Confirmed
TBD	To Be Defined
TBU	To Be Updated
TBW	To Be Written
TC	TeleCommand
TM	TeleMetry
TUVIE	Technical UVIE
UR	User Requirement
URD	User Requirements Document
UVIE	University of Vienna
WE	Warm Electronics

Definition	**Description**
BBID	Building Block Identification
BSID	Bolometer Setup Identification
CDH	Compressed DEC/MEC Header

CDHS	Compressed DEC/MEC Header Size
CEH	Compressed Entity Header
CI	Counter Increments
CMM	Compression Mode
CPR	Chopper Position Readback
CR	Compression Ratio achieved
CRCRMP	Current Readout Count in a RaMP
CRDC	Current ReaDout Count
CRDCCP	Current ReaDout Count in Chopper Position
CRECR	CRE Control Readback
CSD	Compressed Science Data
CSW	Compression SoftWare
DBID	Data Block ID
DECID	DEcompression Code IDentification
DXS ID	Detectors Selection table IDentification
GPR	Grating Position Readback
HK	HouseKeeping
LBL	Label
MEM_STATUS_CNTS	MEMory STATUS CouNTerS
NACK	Negative Acknowledgement
OBSID	Observation Identification
PACK	Positive Acknowledgement
PIX	Packet Index
RCS	Raw Channel Selection
RCX	Raw Channel IndeX
REAL	REduction ALgorithm
ROSP	Readout Specifications in Photometry
ROSS	Readout Specifications in Spectroscopy
RRR	Readouts in Ramp Readback
SCIS	compressed SCIence data Size
SPR	Spare
SPUID	SPU ID
TMP	Timing Parameters
VID	Version ID
VLD	Validity
WPR	Wheel Position Readback

To identify all software requirements they are structured as follows:

ID-type.number

Where

ID	System identification (SPU-OBS for On Board Software User Requirements or SPU-HSR for HLSW Requirements)
type	type of the requirement

… for ID = SPU-HSR:

FN.................Functional

IF ....Interface

OP.................Operational

RS ....Resource

VF.................Verification

AT ....Acceptance Testing

DO ....Documentation

SA.................Safety

QA ....Quality

… for ID = SPU-OBS:

ON ....Switch On

OF.................Switch Off

SW................Software

CR.................Communication

TM................Telemetry

number	serial number of the requirement

1.3 References

1.3.1 Applicable Documents

AD001	PT-PACS-02126	FIRST/PLANCK Instrument Interface Document - Part B – Instrument PACS
AD002	SCI-PT-ICD-7527	Packet Structure Interface Control Document
AD003	PACS-ME-ID-001	PACS Instrument Interface Requirement Document
AD004	PACS-ME-SP-001	The Photoconductor Array Camera & Spectrometer (PACS) for the Far InfraRed and Submillimetre Telescope (FIRST)
AD005	ESA-TM-06	Handbook of Data Compression Algorithms
AD006	PACS-ME-RS-005	PACS Instrument Requirement Document
AD007	IFSI/OBS/PL/2000-001	DPU/ICU On Board Software Product Assurance Plan

1.3.2 Reference Documents

RD001	BSSC(96)2	Guide to applying the ESA software engineering standards to small software projects
RD002	PACS-ME-PL-006	PACS Project Configuration Management Plan
RD003	SCI-PT-IIDA-04624	FIRST/PLANCK Instrument Interface Document IID - Part A -
RD004	PACS-ME-PL-005	Operating Modes of the PACS Instrument
RD005	DIPSAPII-DAS-31-06	SMCS332 User Manual Issue2
RD006	PACS-TW-SR-001	PACS SPU HLSW User Requirement Document
RD007	PACS-TW-ID-001	PACS SPU HLSW to DPU Interface Description
RD008	PACS-IC-RD-001	PACS SPU Start-Up SW and LLSW Drivers URD
RD009	PACS-CL-ID-004	DEC/MEC to SPU Interface Description
RD010	FPL-IC-1214-01-CRS	PACS SPU HW-SW Interface Control Document
RD011	FPL-IC-1214-04-CRS	PACS SPU LLSW drivers SW ICD
RD012	FPL-TN-1214-03-CRS	Planck/Herschel PACS SPU Technical Description (17. July 2001, issue 2)
RD013	Analog Devices	ADSP-21020 User's Manual. Second Edition, USA 1995
RD014	Com. ACM 24 Jun. 1981	Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography
RD015	FPL-SDD-1214-01-CRS	HERSHEL PACS Start up and LLSW Drivers SDD (issue 1)
RD016	PACS-TW-HM-002	PACS SPU HLSW User Manual
RD017	PACS-TW-TS-004	PACS SPU HLSW Acceptance Test Plan for FM Delivery
RD018	Proc. of SPIE Vol. 5487 (2004), Part 1, p. 481	"Herschel/PACS On-noard Reduction/Compression Software Implementation" (by R. Ottensamer et al.)
RD019	PACS-CR-UM-024	DPU OBS User Manual
RD020	PACS-TW-TN-014	The flightdata file from the compression point of view

1.4 Overview of the document

This document is structured as follows:

- Section 1, an introduction and an outline of this document is given.
- Section 2 contains the logical model description of the SPU High Level Software.
- In Section 3, the SW requirements are listed.
- Section 4, a detailed description of the SPU HLSW design is given.
- Section 5 contains details about the SW components and their functions.
- Section 6 includes the summarisation of the computer resources required to build, operate and maintain the software.
- Section 7 contains the User Requirements vs. Software Traceability Matrix
- Section 8 includes the Software Requirements vs. Components Traceability Matrix and
- Section 9 displays the Software Requirements vs. Software Verification & Validation Traceability Matrix

2 Model Description

2.1 General

The Herschel Space Observatory[1] is the fourth cornerstone mission of the European Space Agency (ESA) `Horizon 2000' science plan. It will be implemented together with the Planck mission as a single project. It is defined as a multi-user "observatory type" mission with the goal to open up the wavelength range ≤ 60-600 µm to photometry and spectroscopy with unprecedented sensitivity and spatial resolution, unobscured by the Earth's atmosphere. The Herschel space observatory will be launched in 2008 to operate at a distance of 1.5 million km far from the earth. The scientific topics of the mission include the formation and evolution of galaxies in the early universe, the search for protostars in our own Galaxy, and the evolution of planetary systems including the Solar System.

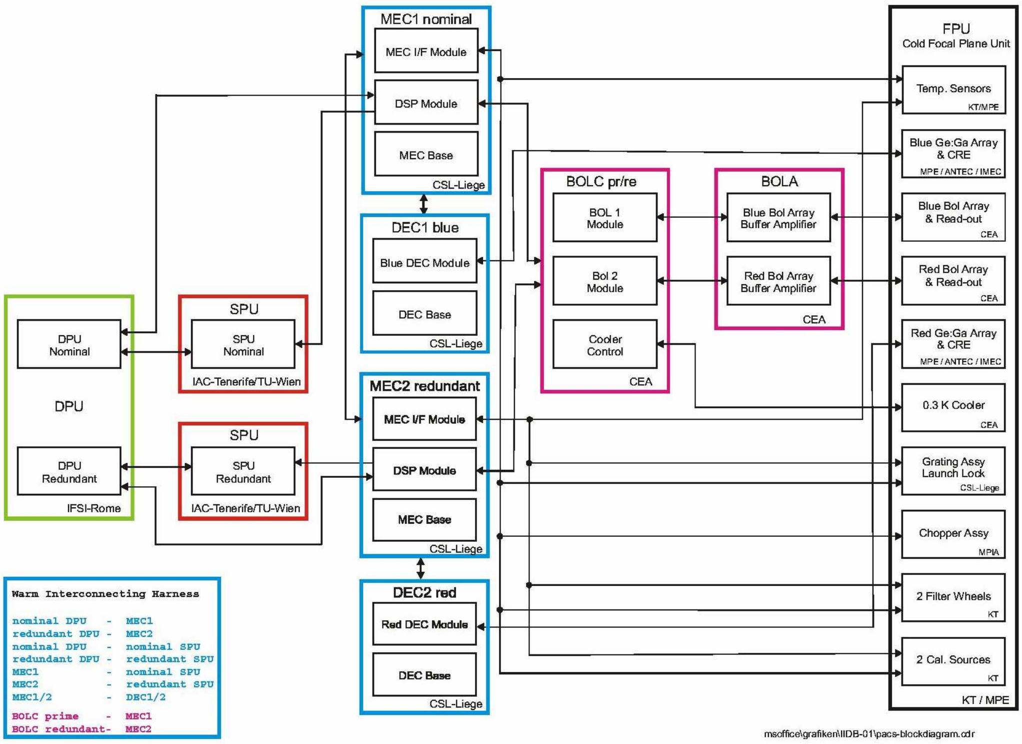


Figure 1. PACS Instrument Block Diagram

[1] The previous name of Herschel Space Observatory was Far InfraRed and Submillimeter Telescope 'FIRST'

A payload complement of three instruments is housed inside a super fluid helium cryostat of the Telescope: "HIFI", "SPIRE" and "PACS". Three consortia develop them. Many of the involved processes are known to emit most of their luminosity in the far-infrared and sub-millimetre band, both as continuum radiation from dust and as spectral line features. The shortest wavelength band, 60-210 μm, will be covered by the Photoconductor Array Camera & Spectrometer (PACS), which will provide both photometric and spectroscopic observing modes suited to address the key scientific topics of the HERSCHEL mission. The electrical, thermal and configuration aspect of the PACS instrument is given in *Figure 1*. As shown in this diagram, the instrument contains several sub-units, which are developed by different institutions in a consortium led by MPE, Garching, Germany.

DEC/MEC, SPU, and DPU represent PACS warm electronic sub-units. CSL, IAC/UVIE and IFSI respectively have the task to develop these modules. They are responsible for data handling and processing according to ground instructions.

2.1.1 The Signal Processing Unit (SPU)

The main function of the Signal Processing Unit (SPU) of the PACS instrument consists of the data reduction and compression for transmission purpose. This task is achieved by the ASW and related programs, implemented on the DSPs. Furthermore, the SPU has to control the communication functionality with DEC/MEC and DPU. *Figure 2* represents the communication links between the SPU unit, DEC/MEC and DPU. Transmission of science data from DEC/MEC is ensured by two IEEE 1355/spacewire data links. Two links also lie between SPU and DPU. They will allow the transmission of compressed data, HLSW HK and DPU/SPU commands/responses. The SPU development consists of two main parts: SPU HW+LLSW and SPU HLSW. We only consider the HLSW in this document.

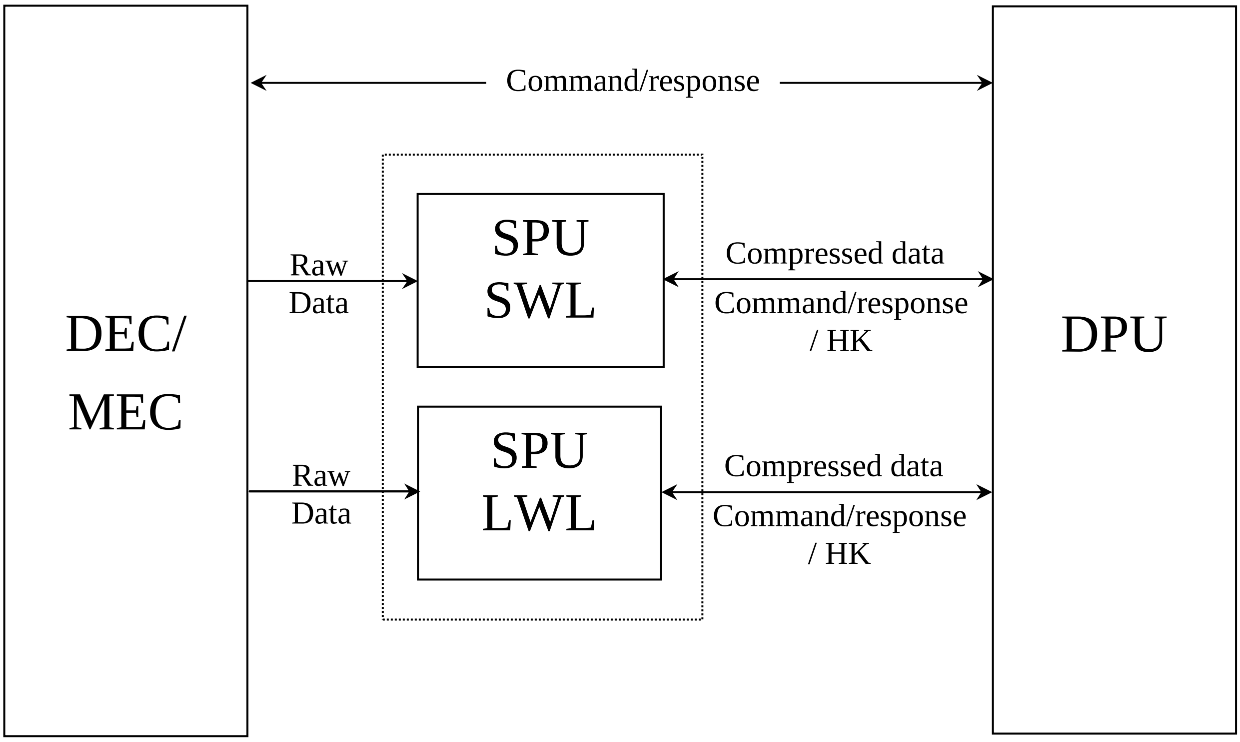


Figure 2. Communication Links between SPU and other Subsystems

The SPU unit consists of two CPU boards, which work independently. They are called SWL SPU and LWL SPU. The HLSW will be identically implemented in each SPU sub-unit (SWL and LWL). As both CPUs have equivalent functionality, we only focus on the HLSW description of one board. The SPU contains two data inputs:

- Science data + header from DEC/MEC
- Commands from DPU

and three data outputs:

- Compressed data to DPU
- HK to DPU
- Command response to DPU

The HLSW operates with these inputs to provide the required output data. The overall DEC/MEC raw data description can be found in the specific document RD009.

2.1.2 The SPU High Level Software (HLSW)

The detector readouts are collected by the DEC/MEC and put into packets. A header is attached to each packet including the data type, the observation configuration and the compression parameters. Then data packets and headers are sent to SPU. The science data are reduced and compressed by the SPU HLSW regarding the DEC/MEC header information. The DEC/MEC header parameters will also be compressed and packaged with the compressed science data and will be sent to the DPU. The HK parameters, which consist either of data reduction-compression results of the HLSW whenever raw data are received from DEC/MEC or the SPU status while no observation is performed, will also be sent to DPU. The compression mode, which represents the way the reduction and compression should be done, is read from the DEC/MEC header. It defines the way the compression should be done.

The HLSW deals with the instrument configuration through the DEC/MEC header information as specified in Section 3.4.2. The PACS instrument configuration consists of the instrument operating and observing modes described in the document RD004.

- **Instrument Operating Modes**: They represent the basic activities carried out within the PACS instrument.
- **Instrument Observing Modes**: They represent the kind of imaging performed by the PACS instrument. The data stream, size, and behaviour will depend on the mode used.

2.2 Logical Model

The SPU HLSW is developed according to ESA engineering software standards described in RD001. The definition of the functions and procedures to establish the software assurance for the SPU HLSW is written in AD007 (approved Product Assurance Plan from DPU SW). The software design methodology is the Yourdon-deMarco structured analysis method with the Hatley-Phirbai real-time extensions.

The interaction between the HLSW, SUSW and LLSW as well as the handover to the SPU HLSW is given in RD015. The general layout of the SPU HLSW is represented in *Figure 3*.

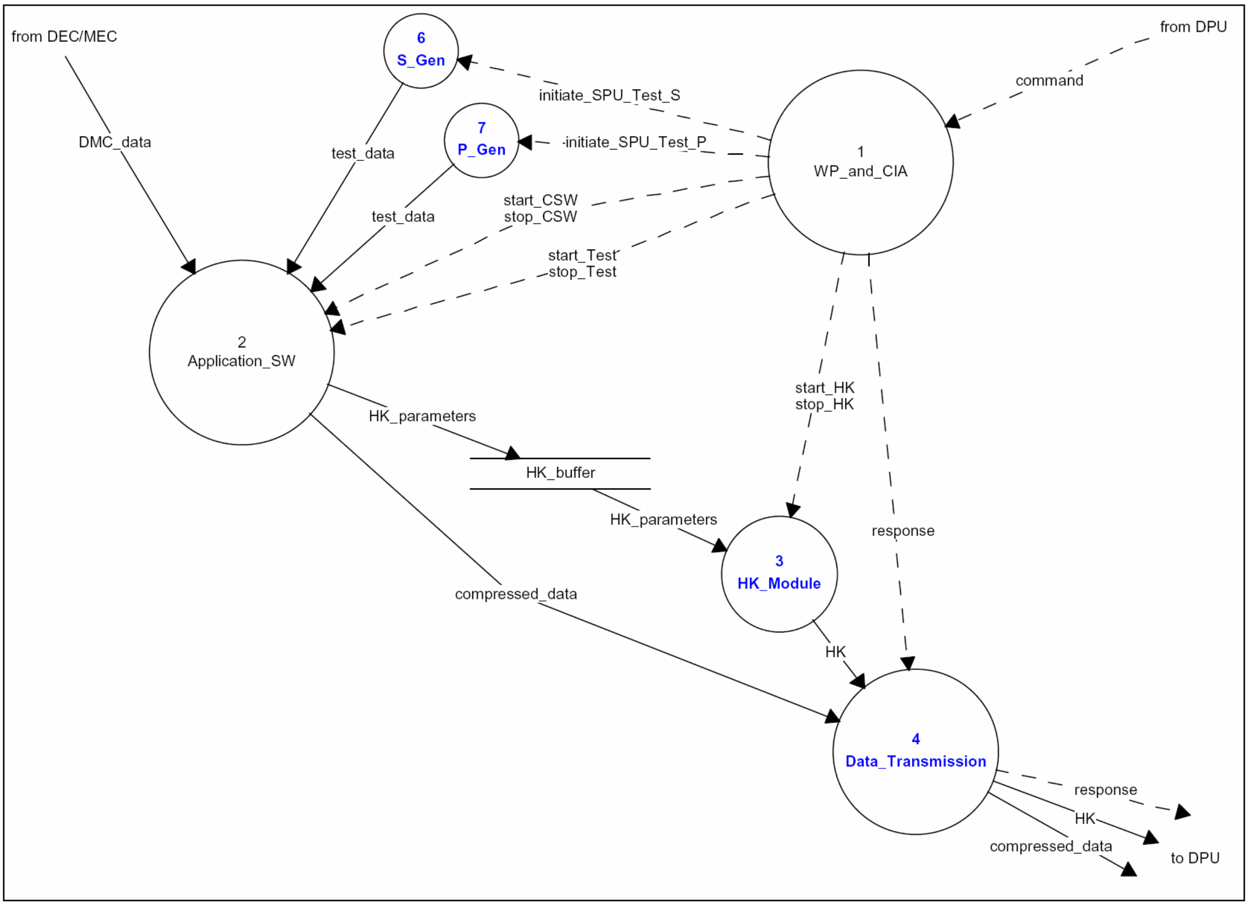


Figure 3. SPU HLSW Architecture

The WP_and_CIA (Watch Process and Command Interpretation and Acknowledgment) module is depicted in Figure 4.

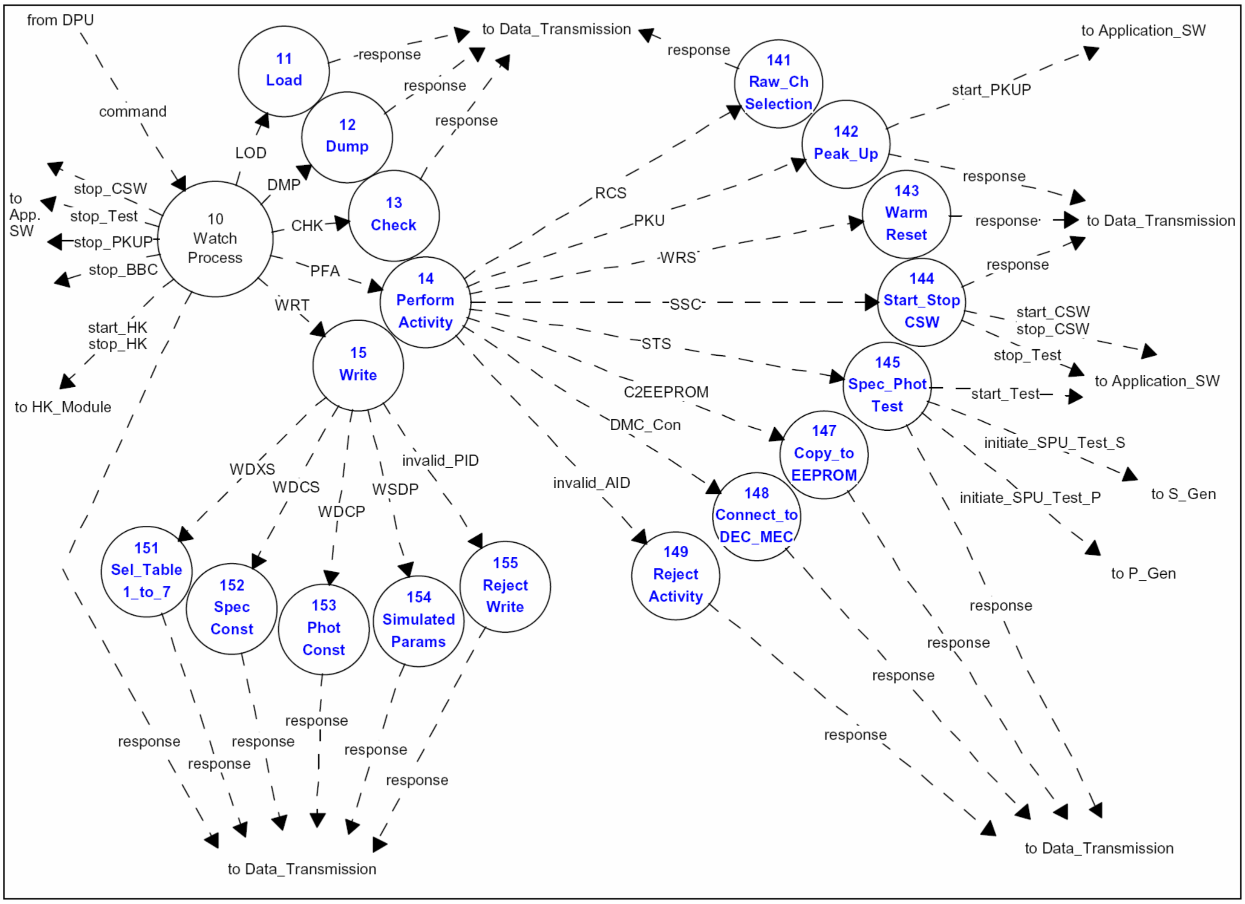


Figure 4. WP_and_CIA Module

The Watch Process module is illustrated in Figure 5.

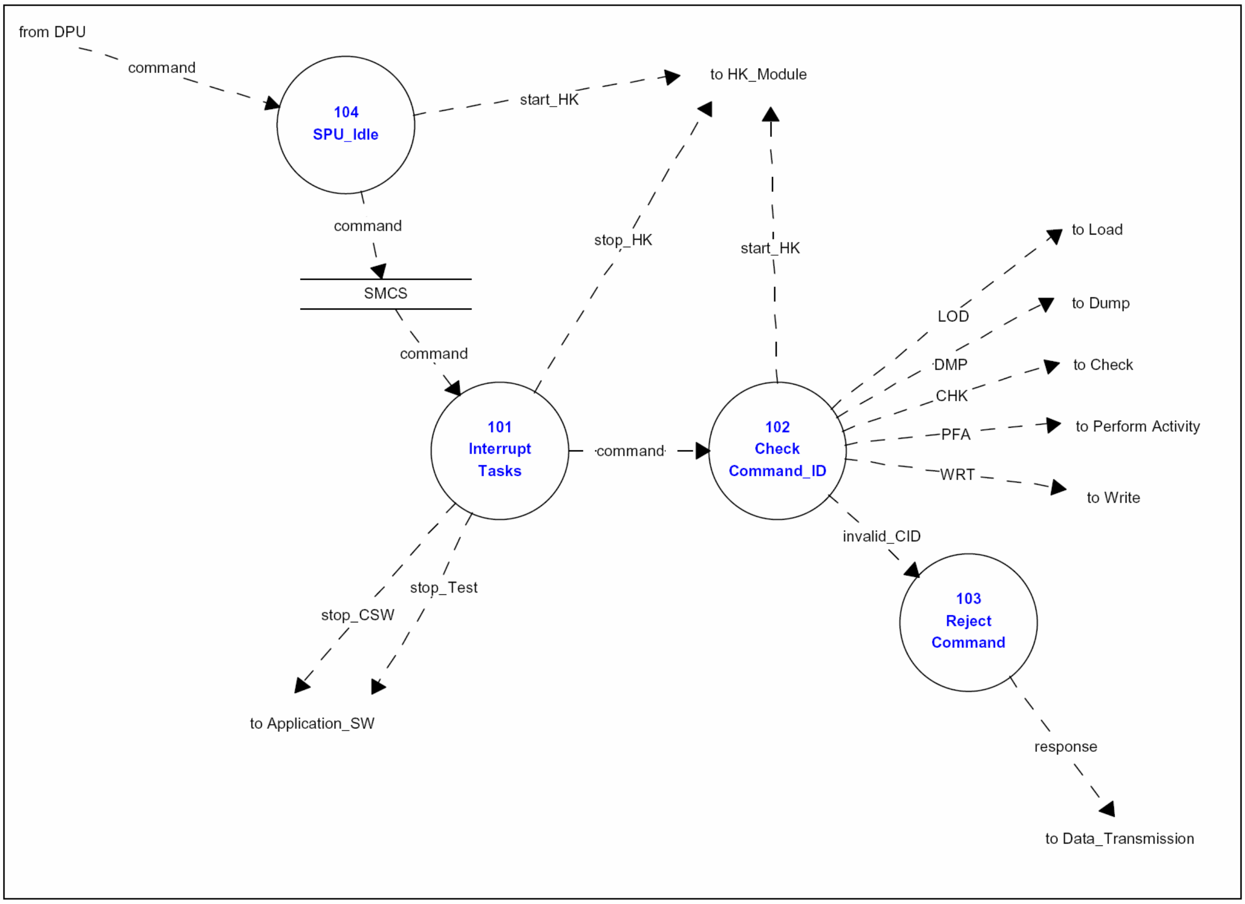


Figure 5. Watch Process Module

The components of the ASW (Application Software) module are shown in Figure 6.

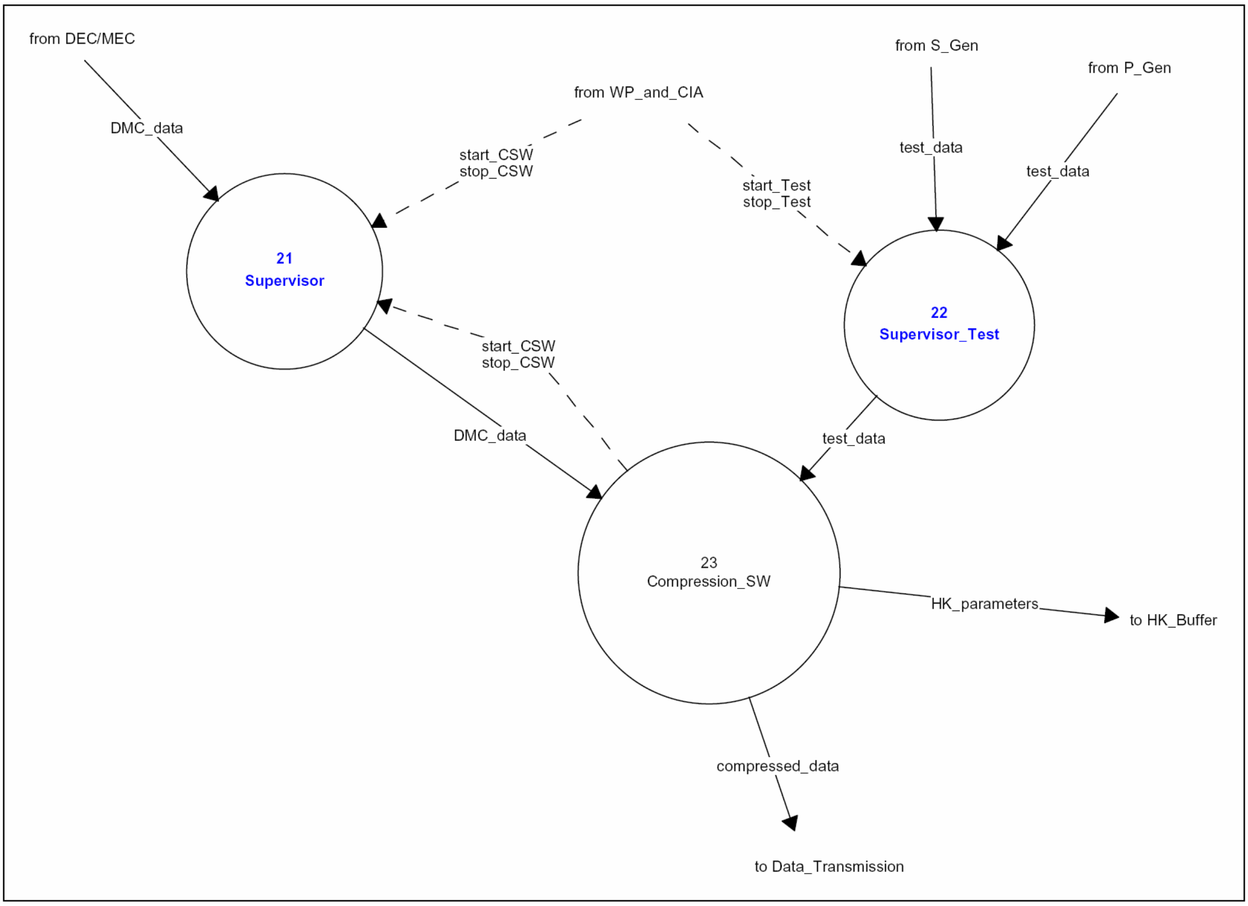


Figure 6. ASW Module

The CSW (Compression Software) concept is given in Figure 7.

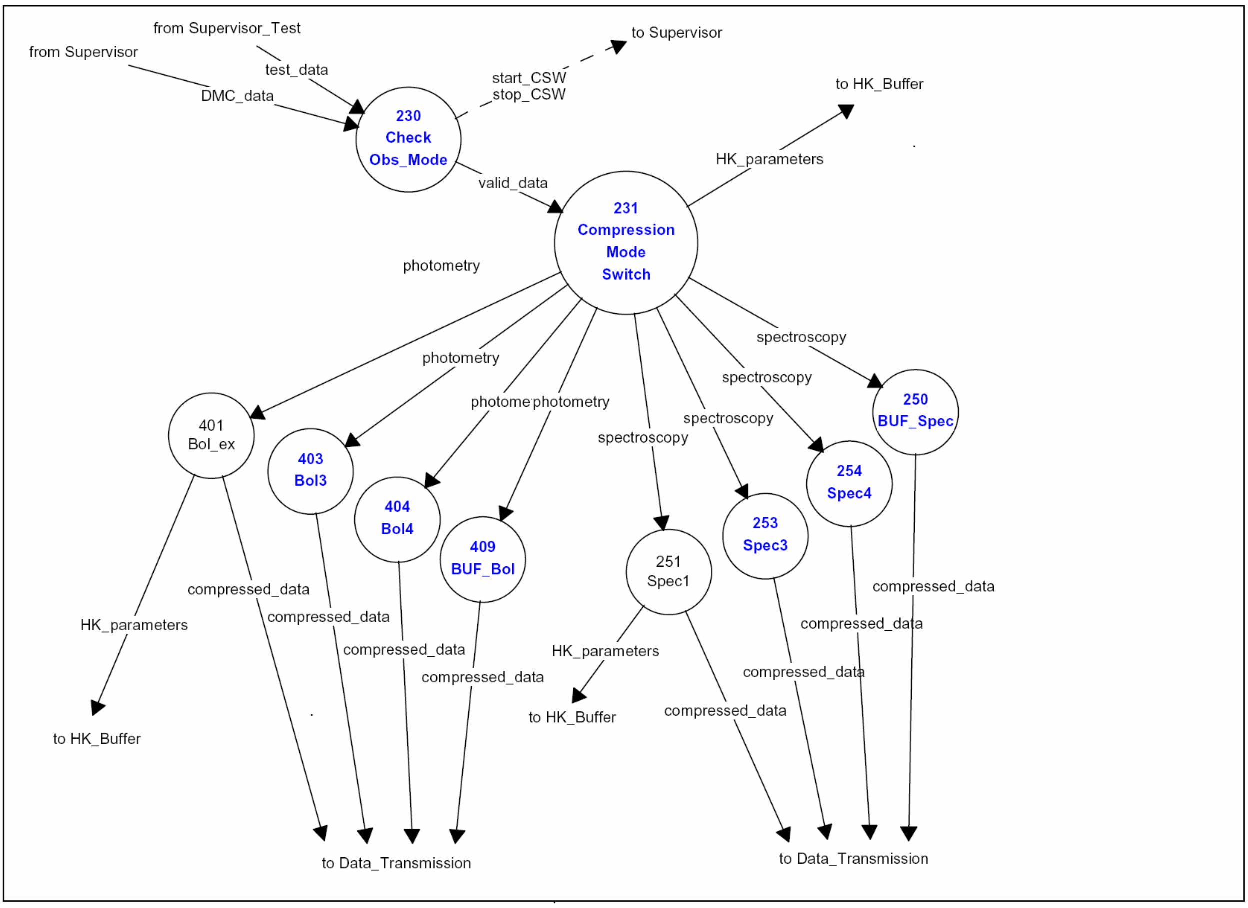


Figure 7. CSW Module

The modules for all HLSW compression modes for Spectroscopy (Spec1 to Spec5 and BUF_Spec) and for Photometry (Bol1 to Bol4 and BUF_Bol) are represented in Figure 8 to .

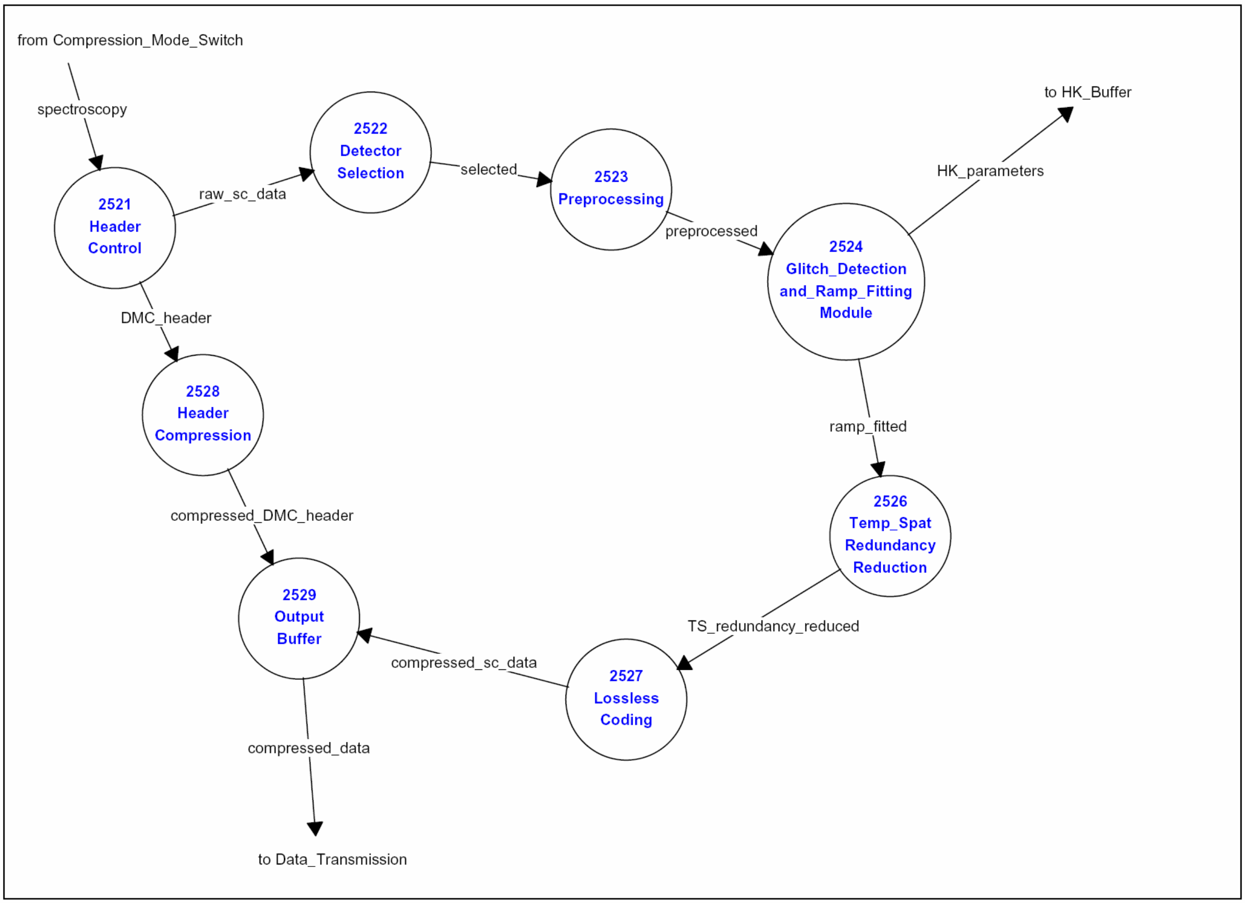


Figure 8. Components of Spectroscopy Mode 1 (Spec1) Module

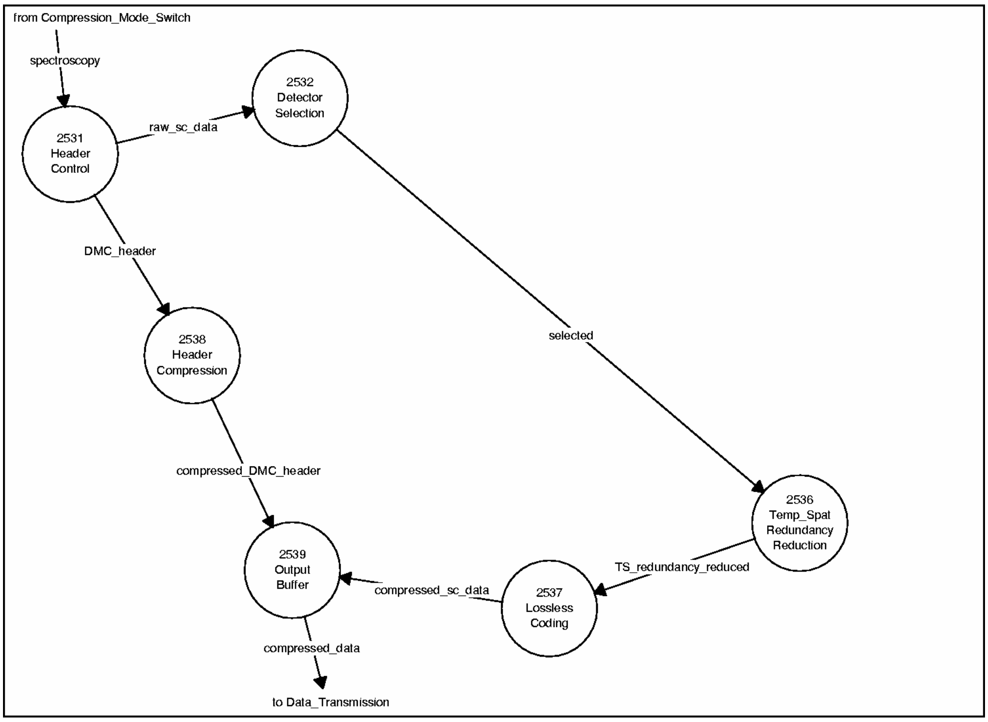


Figure 9. Components of Spectroscopy Mode 3 (Spec3) Module

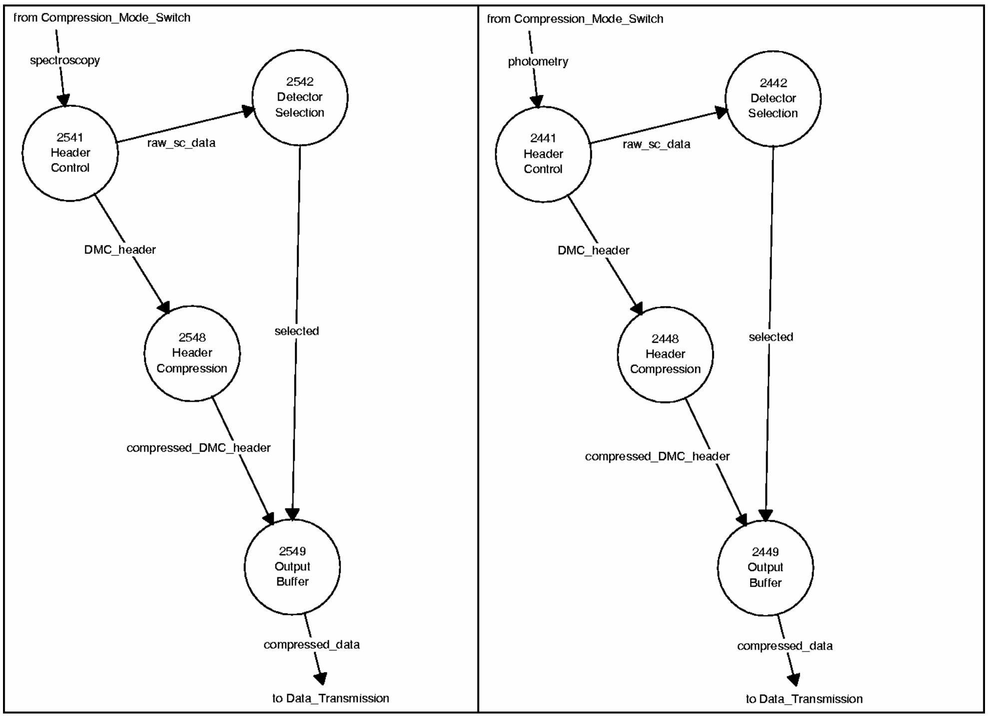


Figure 10. Components of Spectroscopy Mode 4 (Spec4) and Photometry Mode 4 (Bol4) Module

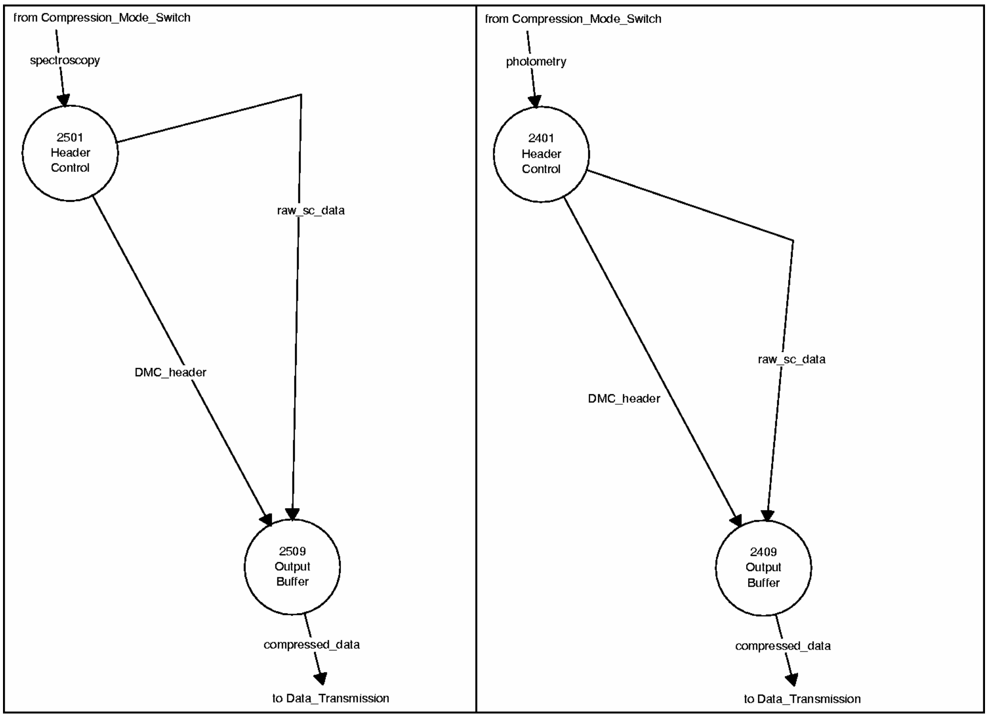


Figure 11. Buffer Transmission Mode in Spectroscopy (BUF_Spec) and Photometry (BUF_Bol)

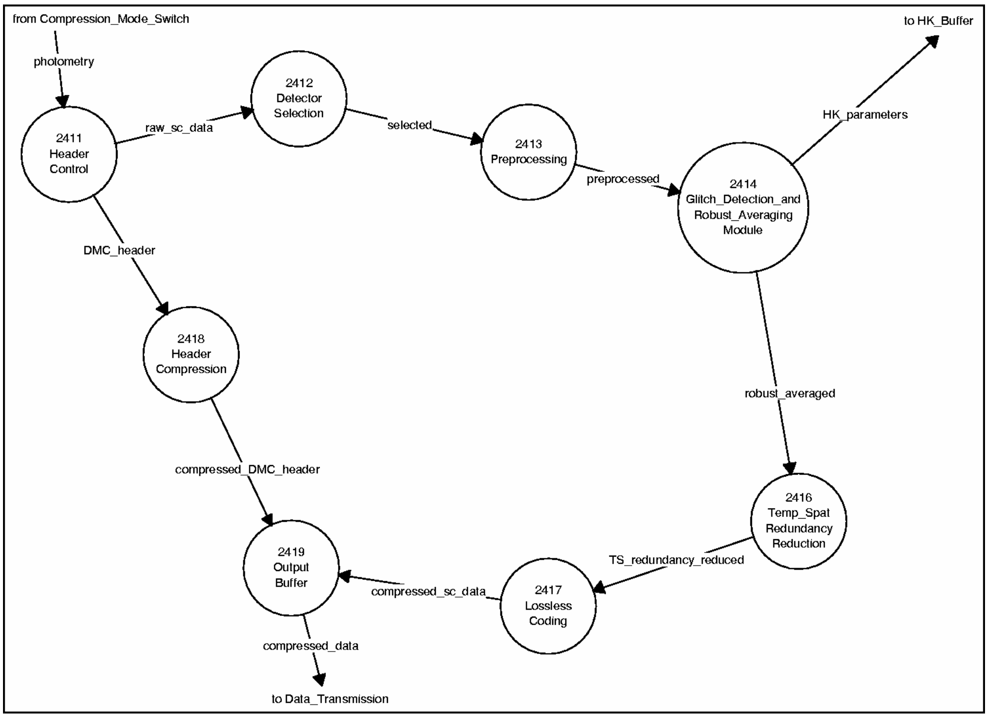


Figure 12. Components of Photometry Mode (Bol_ex) Module

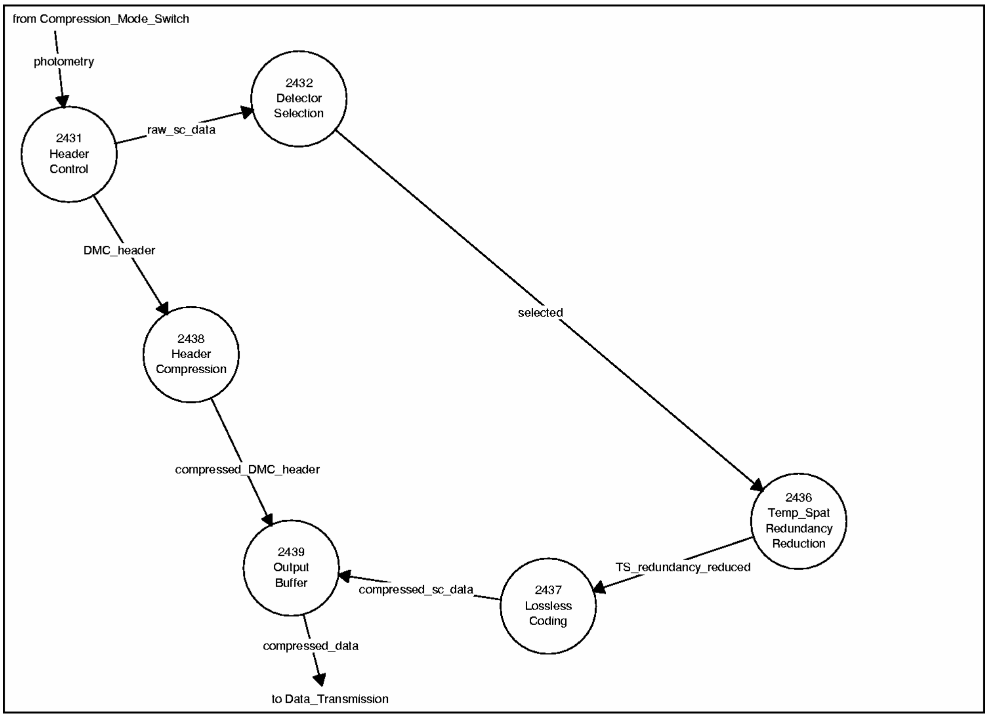


Figure 13. Components of Photometry Mode 3 (Bol3) Module

3 Specific Requirements

3.1 Functional Requirements

3.1.1 The Watch Process

SPU-HSR-FN0001

If the Watch Process detects a received DPU command, it shall read it.

SPU-HSR-FN0002

The Watch Process shall check whether the received DPU command is Load, Dump, Check, Perform, Write or Invalid command by use of reference table.

SPU-HSR-FN0003

The Watch Process shall abort data compression for any received command.

SPU-HSR-FN0004

If the received command is a Load command, attached data shall be checked (via checksum calculation) and written into memory if checksum is right. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0005

The correctness of the Load shall be verified by checking (via checksum calculation) the data written into memory. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0007

If the received command is a Dump command, the required length of data is read from memory. A checksum over the read data is calculated and attached to the dumped data.

SPU-HSR-FN0009

If the received command is a Check command, the required length of memory is checked via the checksum calculation

SPU-HSR-FN0010

Program RAM, Data RAM, Extended Data RAM and Dual Port RAM can be checked by HLSW. The HLSW does not check the EEPROM and the PROM.

SPU-HSR-FN0011

If the received command is a Perform command, the validity of the attached activity shall be checked using a reference table.

SPU-HSR-FN0012

If Warm Reset command is received, HLSW shall clean the data buffers (Input Data buffer, Processing buffer, Compressed DMC Header buffer and RC data buffer), reset the counters (CI and PIX) to zero and reset the connection to DPU and DEC/MEC (reset SMCS chip).

SPU-HSR-FN0013

If Raw Channel Selection command is received, HLSW shall upgrade the Raw Channel Selection table with the attached parameters. The default value of Raw Channel Selection components at HLSW start-up is zero (no selected channel and default channel index is zero).

SPU-HSR-FN0014

If Stop command is received, Watch Process shall signal the abortion of the data compression. If there is no data compression when a Stop command is received no action will be performed.

SPU-HSR-FN0015

If Start command is received, Watch Process shall signal the begin of the data compression. If Start command is received when data compression is ongoing the data compression will be aborted (compound of Stop-Start command).

SPU-HSR-FN0016

If SW Test command for photometry is received, Watch Process shall disable the DEC/MEC link interrupt, enables the internal data generation through Phot_gn and then signal the begin of data compression.

SPU-HSR-FN0017

If SW Test command for spectroscopy is received, Watch Process shall disable the DEC/MEC link interrupt, enables the internal data generation through Spec_gn and then signal the begin of data compression.

SPU-HSR-FN0020

If the received command is a Write command, the validity of the attached parameters shall be checked using a reference table. The write command only checks for valid Parameter ID, valid length and valid checksum, but never checks the content of the attached data. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0021

If the received command is a Write DXS1 (Blue Bolometer sub-array1) command, attached data shall be checked (via checksum calculation). If checksum is correct, HLSW shall upgrade the Table DXS1 with the attached data. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0022

If the received command is a Write DXS2 (Blue Bolometer sub-array2) command, attached data shall be checked (via checksum calculation). If checksum is correct, HLSW shall upgrade the Table DXS2 with the attached data. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0023

If the received command is a Write DXS3 (Blue Bolometer sub-array3) command, attached data shall be checked (via checksum calculation). If checksum is correct, HLSW shall upgrade the Table DXS3 with the attached data. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0024

If the received command is a Write DXS4 (Blue Bolometer sub-array4) command, attached data shall be checked (via checksum calculation). If checksum is correct, HLSW shall upgrade the Table DXS4 with the attached data. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0025

If the received command is a Write DXS5 (Red Bolometer) command, attached data shall be checked (via checksum calculation). If checksum is correct, HLSW shall upgrade the Table DXS5 with the attached data. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0026

If the received command is a Write DXS6 (Photoconductor main array “Blue array for Blue SPU or Red Array for Red SPU”) command, attached data shall be checked (via checksum calculation). If checksum is correct, HLSW shall upgrade the Table DXS6 with the attached data. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0027

If the received command is a Write DXS7 (Photoconductor secondary array “Red array for Blue SPU or Blue Array for Red SPU”) command, attached data shall be checked (via checksum calculation). If checksum is correct, HLSW shall upgrade the Table DXS7 with the attached data. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0028

If the received command is a Write Detector Constants for Photometry command, attached data shall be checked (via checksum calculation). If checksum is correct, HLSW shall upgrade the Detector Constants for Photometry Table with the attached data. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0029

If the received command is a Write Detector Constants for Spectroscopy command, attached data shall be checked (via checksum calculation). If checksum is correct, HLSW shall upgrade the Detector Constants for Spectroscopy Table with the attached data. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0030

If the received command is a Write Simulated Data command, attached data shall be checked (via checksum calculation). If checksum is correct, HLSW shall upgrade the Simulated Data Table with the attached data. If the checksum is wrong a NACK will be signalled.

SPU-HSR-FN0031

If Copy Data from RAM to EEPROM command is received, the HLSW shall copy the program source code data from RAM to EEPROM.

SPU-HSR-FN0032

If Connect to DEC/MEC command is received, the HLSW should start the link to DEC/MEC as slave or master depending on the attached parameter.

3.1.2 Application Software

SPU-HSR-FN0100

If the Application SW detects DEC/MEC data, it shall read them.

SPU-HSR-FN0101

The Application SW shall be able to trigger the Compression Software in case of the START command.

SPU-HSR-FN0102

In case of HLSW Started Mode (i.e. Start command received), the Application SW shall check whether the received data are photometric, spectroscopic or invalid data.

SPU-HSR-FN0103

After reception of Start command, HLSW buffers data starting from the first sample in ramp “spectroscopy”. Data read up to this sample are ignored except in Buffer Transmission Mode.

SPU-HSR-FN0104

A buffer ready for compression shall contain data with uniform header i.e. valid-flagged data, one OBSID, and one compression mode. In case of non-uniform header (i.e. transition to a new OBSID or compression mode), the data buffers are cleaned and filled with new input data.

SPU-HSR-FN0105

The application software shall be able to process data according to the received compressed mode and make it ready to DPU.

3.1.3 HK Module

SPU-HSR-FN0200

HLSW shall transmit HK packets to DPU at maximum time interval of 2000 ms.

SPU-HSR-FN0201

Successive HK packets shall contain an incremental counter starting from 0.

SPU-HSR-FN0202

EDAC status concerning single error correction and double error detection in RAM is signalled within the HK. Memory check status word is generated in every HK packet in SPU idle mode and for every 256 HK packets in SPU started mode.

SPU-HSR-FN0203

Ground SW has to detect the permanent memory errors by dumping and analyzing the failing addresses reported with EDAC check. Information for dumping the array of failing addresses can be found in RD016.

SPU-HSR-FN204

To correct single errors and avoid additional double errors memory scrubbing shall be performed every minute in idle mode and every 20 min when the SW is started.

3.2 Performance Requirements

SPU-HSR-FN0300

In started mode, HLSW shall process a data buffer and transmit the resulted compressed packets before a new buffer is filled and ready for processing.

SPU-HSR-FN0301

In started mode, HLSW shall not integrate slopes from different chopper plateaus.

3.3 Interface Requirements

3.3.1 DPU Commands

SPU-HSR-IF0001

If a DPU command is received, HLSW shall abort any running activity (data compression).

SPU-HSR-IF0002

The maximum response time for the DPU command is 200 ms. In case of Dump and Check commands, data length shall not exceed 10 KB to fulfil the max response time requirement.

SPU-HSR-IF0003

The HLSW shall acknowledge any received DPU commands (NACK or PACK) according to the protocol described in RD007.

3.3.1.1 LOAD Command

SPU-HSR-IF0010

If a Load command is received, HLSW shall abort any running activity, acknowledge the execution of the command according to the protocol described in RD007.

SPU-HSR-IF0011

Data can only be loaded to Data RAM, Extended Data RAM and Dual Port RAM. The HLSW does not write data into Program RAM or EEPROM.

3.3.1.2 DUMP Command

SPU-HSR-IF0020

If a Dump command is received, HLSW shall abort any running activity, acknowledge the execution of the command according to the protocol described in RD007.

SPU-HSR-IF0021

Data can be dumped from Program RAM, Data RAM, Extended Data RAM and Dual Port RAM. The HLSW does not read the EEPROM and the PROM.

3.3.1.3 CHECK Command

SPU-HSR-IF0030

If a Check command is received, HLSW shall abort any running activity, acknowledge the execution of the command if correct parameters according to the protocol described in RD007.

3.3.1.4 RESET Command

SPU-HSR-IF0050

If a Warm Reset command is received, HLSW shall abort any running activity, acknowledge the reception of the command, and execute the command if correct parameters.

3.3.1.5 RAW_CHAN_TRAN_MODE Command

SPU-HSR-IF0060

If Raw Channel Selection command is received, HLSW shall abort any running activity, acknowledge the execution of the command and update the Raw Channel table.

3.3.1.6 STOP_REDUCT_COMPR Command

SPU-HSR-IF0070

If Stop command is received, HLSW shall acknowledge the reception of the command, and execute the command. If there is no data compression when a Stop command is received no action will be performed.

3.3.1.7 START_REDUCT_COMPR Command

SPU-HSR-IF0080

If Start command is received, HLSW shall abort any running activity, acknowledge the reception of the command, and execute the command if parameters are correct.

3.3.1.8 ACT_TEST_PHOT Command

SPU-HSR-IF0090

If SPU Test command for photometry is received, HLSW shall abort any running activity, acknowledge the reception of the command, and execute the command if parameters are correct.

3.3.1.9 ACT_TEST_SPEC

SPU-HSR-IF0100

If SPU Test command for spectroscopy is received, HLSW shall abort any running activity, acknowledge the reception of the command, and execute the command if parameters are correct.

3.3.1.10 ACT_BOL_BGND_CANCEL Command

SPU-HSR-IF0110 … deleted

If Bolometer Background Cancelling command is received, HLSW shall abort any running activity, acknowledge the reception of the command, and execute the command if correct parameters.

3.3.1.11 Write Commands

SPU-HSR-IF0120

If a Write command is received, HLSW shall abort any running activity, acknowledge the execution of the command.

3.3.1.12 CP_DATA_RAM_EEPROM Command

SPU-HSR-IF0130

If a Copy Data from RAM to EEPROM command is received, the HLSW acknowledges the reception of the command and starts copying data into EEPROM if parameters are correct. If the copy to EEPROM fails, the failure shall be reflected in the HK.

SPU-HSR-IF0131

While data is copied from RAM to EEPROM, no HK packets are transmitted to DPU.

3.3.1.13 CONNECT_DMC Command

SPU-HSR-IF0140

If a Connect to DEC/MEC command is received, HLSW acknowledges the reception of the command and starts the Link to DEC/MEC as Master or Slave according to the command parameter settings. The status of the link connection with DEC/MEC is reported in the HK.

3.3.2 The Compressed Entity

SPU-HSR-IF0200

HLSW transmits TM packets with fixed size of 1012 Bytes. The useful data size in the last TM packet is calculated using this formula:

*Data Size in Last TM Packet = (250 * Total Nb. of TM Packets per Compressed entity) – (7+SCIS+CDMHS)*

The size unit is 4-byte words.

SPU-HSR-IF0201

In started mode, HLSW transmits output packets (TM packets or HK packets) to DPU. HLSW may buffer a maximum number of 75 output packets in a FIFO whenever DPU is not ready to receive packets from SPU. Overflowing this number of packets, TM packets are lost. Remark: the DPU is not informed that the buffer has overflowed. The DPU input buffer is limited to 400 packets (RD019), therefore modes with higher data rates than 120 kbit/s (Buffer Transmission Mode) must send the data delayed.

3.3.3 DEC/MEC to SPU Interface

SPU-HSR-IF0300

HLSW does not acknowledge the reception of DEC/MEC data packets.

SPU-HSR-IF0301

HLSW shall detect corrupted DEC/MEC data packets through the fields: Type and CMM. Packets must be of valid Type and CMM, otherwise they are designated as corrupt.

SPU-HSR-IF0302

HLSW shall distinguish between photometric and spectroscopic data through the Type field.

SPU-HSR-IF0303

The HLSW shall signal the DMC Link connection status in the SPU HK.

SPU-HSR-IF0304

The maximum DEC/MEC packet size accepted by the HLSW is 250 Word in spectroscopy and 272 Word in photometry as specified in RD009. If the received packet lengths differs from these specifications a DEC/MEC header error is signalled in the HK.

3.4 Operational Requirements

3.4.1 DEC/MEC Header Usage

3.4.1.1 Spectroscopy

Generality

SPU-HSR-OP0001 … obsolete (this restriction does no longer apply)

Buffer granularity of the compression SW is 8Hz. i.e. SPU starts data compression if at least 32 frames (256Hz readout rate) are collected for a coherent DEC/MEC header. If not the already stored data frames are deleted and a new buffering will start. Coherent header refers to the label usage (see Label Section)

SPU-HSR-OP0002

SPU starts compression after collecting a maximum number of frames of 512 (2s Buffer at 256Hz readout rate).

SPU-HSR-OP0003

DEC/MEC header is compressed lossless and transmitted within TM.

SPUID

SPU-HSR-OP0010

SPUID is not taken into account. It is deleted from the buffer.

Type

SPU-HSR-OP0020

If the Type is not conform to the specifications (1 for Spectroscopy packet and 2 for Photometry packet), the science data is not buffered and a DEC/MEC header error is signalled in the SPU HK.

SPU-HSR-OP0021

If the Type is set to 1 „Spectroscopy packet", HLSW checks CRCRMP to start buffering with the first sample of a ramp.

Current Readout Count in a Ramp

SPU-HSR-OP0030

This field signals the begin of a ramp. It is the first readout stored in a buffer.

SPU-HSR-OP0031

Each begin of a ramp, chopper position, grating position and validity fields are checked and stored as lookup table for processing this buffer.

Readout in Ramp Readback

SPU-HSR-OP0040

This field together with CRCRMP signals the begin of a ramp.

SPU-HSR-OP0041

The last RRR is used for the compression i.e. if a begin and an end of buffer has different RRR, then ramps are fit using the last received RRR.

Compression parameter

SPU-HSR-OP0050

If a buffer is stored with compression parameter not conform to the specifications, a DEC/MEC header error is signalled in the SPU HK and the buffer is compressed in the default compression mode for spectroscopy.

SPU-HSR-OP0051

If a buffer storage is not completed (Frames collected are less than 512) and the compression parameter has changed, the previous buffer is closed, compressed and transmitted. Then a new buffer storage begins. The last ramp of the previous buffer may be cancelled if the new CMM value arrives within a ramp

Validity

SPU-HSR-OP0070

This field is checked for every ramp once. It is only checked for the first sample of a ramp.

SPU-HSR-OP0071 … obsolete

If several ramps should be integrated, invalid-flagged ramps are not included.

SPU-HSR-OP0072

If no integration over ramps is required on board, invalid ramps are also transmitted within the TM.

Chopper Position Readback

SPU-HSR-OP0080

This field is checked for every ramp once. It is only checked for the first sample of a ramp.

SPU-HSR-OP0081 … obsolete

If several ramps should be integrated, only successive ramps from the same chopper plateau are integrated together. Ramps from different chopper plateaus are not together integrated.

SPU-HSR-OP0082

A slope is calculated over a ramp within chopper motion if not invalid-flagged.

SPU-HSR-OP0083

The default value for the max allowed chopper position deviation is 100. It is calculated as following:

Max.Chopper position deviation = Abs (Actual CPR – Previous CPR)

Label

SPU-HSR-OP0090

Bit 0 of this field is checked. It set to 1, and then the DEC/MEC sequence for wavelength switching mode is running. The grating position is taken into account for the integration. Chopper positions are not taken into account.

SPU-HSR-OP0091

If label is signalled, the previous stored buffer is closed (if bigger than buffer granularity), and individually compressed while a new buffer will begin. If there is no previous stored buffer nothing will be sent and a new buffer will begin.

Grating Position Readback

SPU-HSR-OP0100

This field is checked only in the wavelength-switching mode. It is checked for every ramp once. It is only checked for the first sample of a ramp.

SPU-HSR-OP0101 … obsolete

If several ramps should be integrated, only successive ramps for the same grating position are integrated together. Ramps from different grating positions are not together integrated.

SPU-HSR-OP0102

The default value for the max allowed grating position deviation is 100. It is calculated as following:

Max. grating position deviation = Abs (Actual GPR – Pervious GPR)

BBID

SPU-HSR-OP0110

BBID is not taken into account in the compression. It is transmitted within the TM packet.

Time

SPU-HSR-OP0120

Time is not taken into account in the compression. It is transmitted within the TM packet.

Wheel Position Readback

SPU-HSR-OP0130

WPR is not taken into account in the compression. It is transmitted within the TM packet.

CRECR

SPU-HSR-OP0140

CRECR is not taken into account in the compression. It is transmitted within the TM packet.

3.4.1.2 Photometry

Generality

SPU-HSR-OP0200 … obsolete (no granularity)

Buffer granularity of the compression SW is 10Hz for the Blue SPU and 2.5Hz for the red SPU i.e. SPU starts data compression if at least 4 frames for blue and 16 frames for red SPU (40Hz readout rate) are collected for a coherent DEC/MEC header. If not, the already stored data frames are deleted and a new buffering will starts. Coherent header refers to the label usage (see Label Section)

SPU-HSR-OP0201

SPU starts compression after collecting a maximum number of frames of 120 frames for Blue SPU and 480 frames for red SPU (3s and 12s Buffer at 40Hz readout rate). Compression also starts for less collected number of frames if the label field is signalled (See label section)

SPU-HSR-OP0202

DEC/MEC header is compressed lossless and transmitted within TM.

SPUID

SPU-HSR-OP0210

SPUID is not taken into account. It is deleted from the buffer in the actual version of the SPU HLSW.

Type

SPU-HSR-OP0220

If the Type is set to 2 „Photometry packet“, HLSW checks CRDCCP to start buffering with the first sample in chopper plateau.

Current Readout Count in a Chopper Plateau

SPU-HSR-OP0230

This field signals the begin of a chopper plateau „average“. It is the first readout stored in a buffer.

SPU-HSR-OP0231

Each time this counter is set to 1, chopper position is checked and stored as lookup table for processing this buffer.

Current Readout Count

SPU-HSR-OP0240

This field is not taken into account in the compression. It is transmitted within the TM.

Compression parameter

SPU-HSR-OP0250

If a buffer is stored with compression parameter not conforming to the specifications, a DEC/MEC header error is signalled in the SPU HK and the buffer is compressed in the default compression mode for photometry.

SPU-HSR-OP0251

If a buffer storage is not completed (Frames collected are less than 120 for blue or 480 for red SPU) and the compression parameter has changed, the previous buffer is closed, compressed and transmitted. Then a new buffer storage begins.

Validity

SPU-HSR-OP0270

This field is checked for every readout.

SPU-HSR-OP0271

Invalid-flagged readouts are discarded and the averaging always begins using the first valid sample.

SPU-HSR-OP0272

If all samples for an average are flagged invalid, they will not be transmitted within the TM.

Chopper Position Readback

SPU-HSR-OP0280

If integration, only successive averages from the same chopper plateau are integrated together. Averages from different chopper plateaus are not together integrated. This may affect the compression rate in the parallel mode if chopper is moving faster than 5 Hz.

SPU-HSR-OP0281

The default value for the max allowed chopper position deviation is 100. It is calculated as following:

Max.Chopper position deviation = Abs (Actual CPR – Pervious CPR)

Label

SPU-HSR-OP0290

Bit 0 of this field shall be checked. It indicates the start of a sequence, if set to 1.

SPU-HSR-OP0291

If label is signalled, the previous stored buffer is closed, and individually compressed while a new buffer will begin.

Data Block ID

SPU-HSR-OP0300

This field is used to distinguish between blue and red SPU. An incorrect value will cause a DMC header error signalled in the HK.

BBID

SPU-HSR-OP0310

BBID is not taken into account in the compression. It is transmitted within the TM packet.

Time

SPU-HSR-OP0320

Time is not taken into account in the compression. It is transmitted within the TM packet.

Wheel Position Readback

SPU-HSR-OP0330

WPR is not taken into account in the compression. It is transmitted within the TM packet.

BSID

SPU-HSR-OP0340

BSID is not taken into account in the compression. It is transmitted within the TM packet.

3.4.2 Instrument Configuration

3.4.2.1 Instrument Operating Modes

SPU-HSR-OP0400

The SPU HLSW shall be adapted to operate with any of the instrument operating modes described in Table 1.

Mode	Description	SPU Task
Safe	PACS is in this mode when other instrument are prime	HK transmission
Standby	State needed for the stabilization of the CRE	Compressed data + HK transmission
Prime	All instrument observing modes listed below in *Table 2*	Compressed data + HK transmission
Parallel	All instrument observing modes listed below in *Table 2*	Compressed data + HK transmission
Test	Check instrument functionality with synthetic data	Compressed data + HK transmission

Table 1. PACS Instrument Operating Modes

SPU-HSR-OP0401

The SPU HLSW shall allow a safe transition from a mode to another with a minimum loss of useful scientific data.

SPU-HSR-OP0402

The parameters of the instrument-operating mode are not sent to the SPU HLSW with the DEC/MEC raw data.

3.4.2.2 Instrument Observing Modes

SPU-HSR-OP0500

The SPU HLSW shall be adapted to operate with any of the instrument observing modes described in *Table 2*.

Mode	Description
Dual-Band Photometry	Standard photometry mode. Both detector arrays are operating simultaneously (BOL)
Single-Band Photometry	One detector array is operating (BOL).
Line Spectroscopy	A short spectral coverage for each pixel is produced. Both detector arrays (PHC) are used at a time
Range Spectroscopy	Large spectral range coverage for each pixel is produced. Both detector arrays (PHC) are used at a time
Parallel	Dual-Band and Single-Band Photometry are used
Calibration Measurements/Observations	Various instrumental settings

Table 2. PACS Instrument Observing Modes

SPU-HSR-OP0501

The instrument observing parameters are not sent within the DEC/MEC header to the SPU HLSW.

3.4.3 SPU HLSW Compression Modes

3.4.3.1 Compressed Data Transmission Mode

SPU-HSR-OP0600

The transmission of additional raw data is optional. The remaining difference to the maximum data rate (120 kbits/s) could be used for transmission of lossless-compressed raw data from a few consecutive channels if Raw Channel Selection parameters are set accordingly.

SPU-HSR-OP0601

The DEC/MEC header shall be compressed and transmitted with the compressed data except in the buffer transmission mode.

SPU-HSR-OP0602

The following parameters shall be sent within the SPU HLSW HK: Housekeeping Header, Observation Identification, Packet Index, Counter Increment, Reduction Algorithm used, Saturation Flag, Glitch counter information, Number of maintained subramps, CPU workload, DEC/MEC link status, Number of integrated ramps, SW Version Identification, Raw Channel Index, DEC/MEC header error, Counters for EDAC Memory Checks, SW Sub-version, Invalid LLC Parameter and Monitored Write Parameter.

SPU-HSR-OP0603

In nominal case one compressed entity is generated every 12 seconds for red photometry (LWL) while it is generated at 3 seconds interval for blue photometry (SWL).

SPU-HSR-OP0604

In nominal case one compressed entity is generated every 2 seconds for spectroscopy (LWL and SWL).

3.4.3.1.1 Default Compression Mode

SPU-HSR-OP0610

HLSW shall be able to support the TM rate of 120 Kbits/s in the default compression mode.

SPU-HSR-OP0611

In started compression SW, the transmitted HK packet following an already-sent compression entity shall contain the compression results as described in RD007.

3.4.3.1.2 Double Compression Mode

SPU-HSR-OP0620

HLSW shall be able to support the TM rate of parallel mode by integrating over a predefined set of readouts.

3.4.3.1.3 Half-Compression Mode (Photometry only)

SPU-HSR-OP0625

For 20 Hz readout rate in photometry, every two successive samples are reduced (averaged) in nominal case. One compressed entity is generated every 24 seconds for red photometry (LWL) while it is generated at 6 seconds interval for blue photometry (SWL).

3.4.3.1.4 Lossless Compression Mode

SPU-HSR-OP0630

HLSW shall be able to select a few detectors data through DXS tables, to compress them lossless and to make them available to DPU.

SPU-HSR-OP0631

HLSW shall not compress all selected pixels lossless to avoid SW crash. Maximum data size for lossless compression mode is decribed in Section 5.9.6.

SPU-HSR-OP0632

If all pixels are selected, HLSW shall be able to select few detectors data (Maximum data size) and compress them lossless.

3.4.3.2 SPU Test Mode

SPU-HSR-OP0640

HLSW shall be able to support SPU test mode. It generates internally simulated data and compresses them.

3.4.3.3 Transparent Mode

SPU-HSR-OP0650

HLSW shall be able to support SPU transparent mode with a few selected pixels.

3.4.3.4 Buffer Transmission Mode

SPU-HSR-OP0660

Buffer transmission mode (described in Section 5.8.5 for photometry and Section 5.8.9 for spectroscopy) shall be used in PACS Burst mode. Remark: the DPU is not informed that burst mode is executing.

SPU-HSR-OP0661

Compression software shall be in stopped state before the run of the buffer compression and vice versa. The switch to the buffer transmission mode from an already commanded compression mode, while compression software is started, may lead to an unpredictable result.

3.4.3.5 Compression Mode for 4s Reset (Spectroscopy only)

SPU-HSR-OP0670

The 4 seconds data buffer shall be subdivided to two identical 2 seconds buffers, where each sub-buffer is independently compressed and put into telemetry.

3.5 Resource Requirements

SPU-HSR-RS0001

HLSW shall have 20% more memory capacity at the time of DDR.

SPU-HSR-RS0100

HLSW software shall have 20% spare processing capacity at the time of the DDR.

3.6 Verification Requirements

SPU-HSR-VF0001

The software shall undergo acceptance verification based at the validation and verification test plan at the following levels:

- Equipment level
- Subsystem level
- System level

SPU-HSR-VF0002

It should be ensured that compressed data can be decompressed to yield the original data.

SPU-HSR-VF0003

It should be ensured that test data sets are chosen to fully execute all paths of the algorithm logic, and to demonstrate handling of worst-case data.

3.7 Acceptance Testing Requirements

SPU-HSR-AT0001

For Acceptance Tests the test cases listed in the following table shall be performed.

Task name	AVM	CQM	PFM	FS
DPU to SPU SWL Communication Interface				
SWTC.1	X	X	X	X
DEC/MEC to SPU SWL Communication Interface				
SWTC.2	X	X	X	X
Application Software Test				
SWTC.3	X	X	X	X
Integration of HLSW in LWL SPU				
SWTC.4	X	X	X	X

Table 3, Acceptance Test

3.8 Documentation Requirements

SPU-HSR-DO0001

The software specification document shall replace the three documents: Architectural Design Document, Detailed Design Document and the Software Requirement Document.

3.9 Portability Requirements

SPU-HSR-PT0001

The software shall be compiled with G21K from Analog Devices. It is runnable on the ADSP 21020 with the communication chip SMCS332 and under Virtuoso Operating System.

3.10 Quality Requirements

SPU-HSR-QA0001

The software development shall be performed in compliance with the Software Quality Assurance Plan (see AD007).

SPU-HSR-QA0002

The software source code shall be commented according to the rules described in AD007.

3.11 Reliability Requirements

SPU-HSR-QA0100

The software shall handle anomalies with respect to:

- Command execution

SPU-HSR-QA0101

The software shall be designed to avoid

- deadlock, starvation and endless looping.
- dynamic memory allocation.

SPU-HSR-QA0102

The software shall include mechanisms in respect to protect data and resources shared between processes.

SPU-HSR-QA0103

The software shall be able to handle RAM and EEPROM SEU.

SPU-HSR-QA0104

The software shall be able to handle corrupt and incomplete DEC/MEC data.

SPU-HSR-QA0105

The software shall be able to detect and handle arithmetic errors. Divisions by 0 shall yield to the result 0xFFFFFFFF (-1).

SPU-HSR-QA0106

The software shall not lead to buffer overflow/underrun.

3.12 Maintainability Requirements

SPU-HSR-QA0200

The software shall be maintainable to later extent.

SPU-HSR-QA0201

Maintainability shall contain error detection and fault repair until launch.

SPU-HSR-QA0202

The SW build shall follow automated procedures using deliverable scripts.

SPU-HSR-QA0203

The SDE shall include all software licences and tools to be frozen and made available to MPE.

SPU-HSR-QA0204

The SW deliveries shall include a detailed map file produced by ADSP compiler.

SPU-HSR-QA0205

The in-house tools needed in development or test shall be documented.

SPU-HSR-QA0206

For problem reporting the SPR/SCR system at http://pacs.ster.kuleuven.ac.be/ shall be used and for CVS repository management the CVS system at http://cvs.ster.kuleuven.be/ shall be used.

3.13 Safety Requirements

SPU-HSR-SA0001

The order of commands, HK and telemetry are classified as safety critical. The commanding must strictly follow the User Manual (RD016).

SPU-HSR-SA0010

The HK shall provide a detailed SPU HLSW status, which should indicate memory failures on the SPU or data consistency failures (DEC/MEC Header).

SPU-HSR-SA0011

Memory failures detected by the EDAC should be indicated in the HK The corresponding HK parameter is described in RD016. Permanent memory errors should be isolated and recovered by dumping and analyzing the failing addresses.

SPU-HSR-SA0012

The HLSW shall signal DEC/MEC header errors in the HK. The list of errors is given in RD016.

4 System Design

4.1 Design Method

A top down approach is used for structuring the HLSW in small modules.

4.2 Decomposition Description

The HLSW has been decomposed in three main tasks

- The **Watch Process**: This task listens to the DPU link. It interrupts the Application SW whenever a command is received for its acknowledgment. Basically, any running activity on board SPU is interrupted whenever a command is received from DPU.

- The **Application Software**: This task performs data reduction and compression according to DEC/MEC header received within the raw data. Its responsibility is to achieve the required compression ratio according to telemetry requirements and to the compression mode.

- The **HK module**: This task generates HK, which is sent to the DPU at 1.9 seconds interval.

A detailed component description follows in Section 5.

4.2.1 The Watch Process and Command Interpretation and Acknowledgement

The responsibility of this module is to listen to the DPU communication link. If a DPU command is received, actual running activities are interrupted and the command is acknowledged and executed. In fact, the DPU command consist of memory load, dump, check, write or perform the sub-unit warm reset, start/stop compression, start SPU test mode and raw channel selection. The description of these commands is reported in RD007.

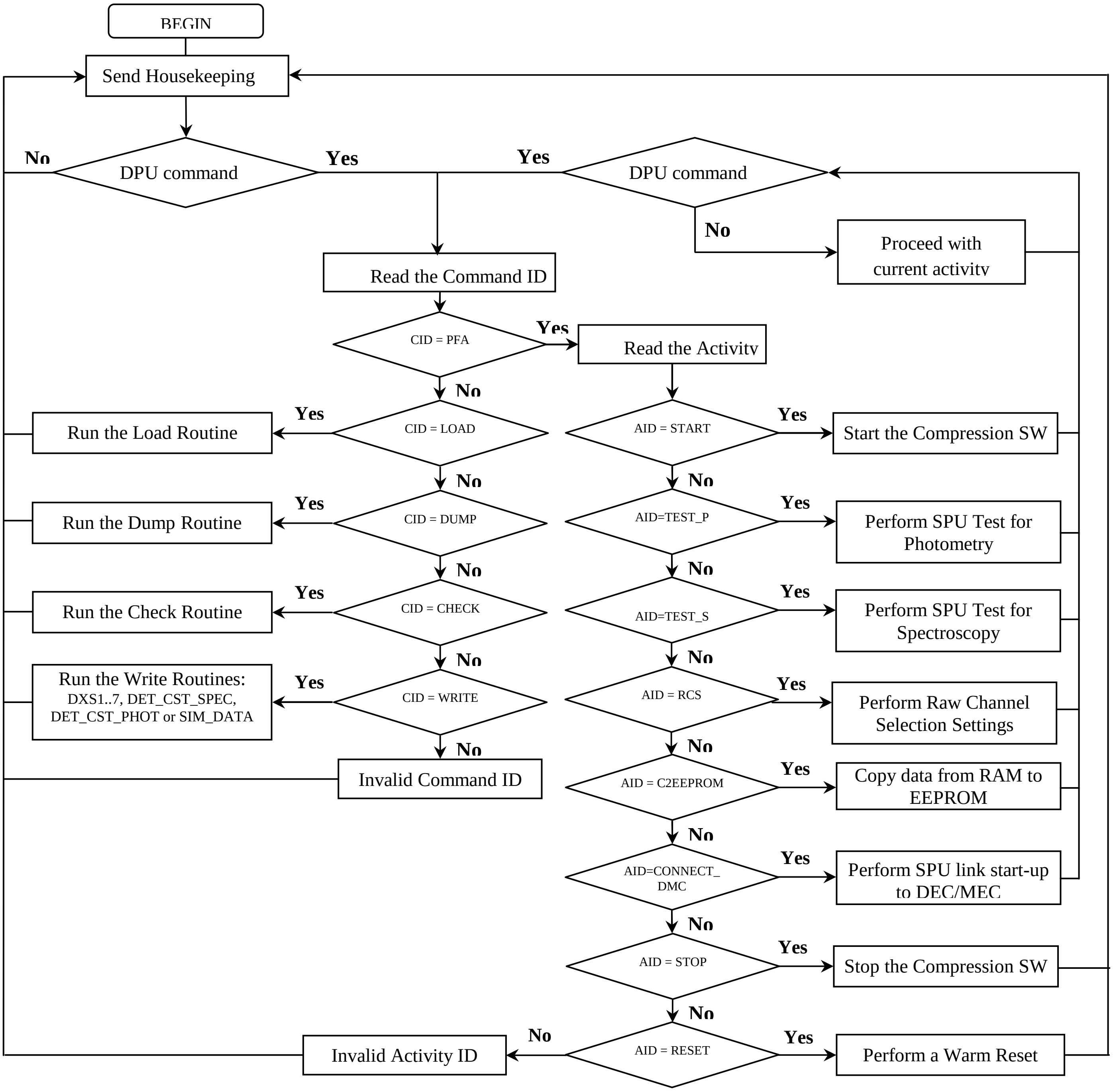


Figure 14. Flowchart of the Watch Process + Command Acknowledgment Algorithm

4.2.2 The Application SW

This task performs data reduction and compression according to the DEC/MEC header. The ASW consists of 2 main parts:

- Supervisor
- Compression software

4.2.2.1 Supervisor

The DEC/MEC science data + header are composed of scientific data and information about the observation configuration and the compression parameters.

The supervisor is implemented at the top of the SPU ASW. It reads from the DEC/MEC header the way the compression should be done. Then, it buffers the separately DEC/MEC header and the science data. Finally, it activates the compression software whenever a buffer is ready for compression.

Furthermore, the supervisor listens to the watch process and wait for a signal to start or stop data processing.

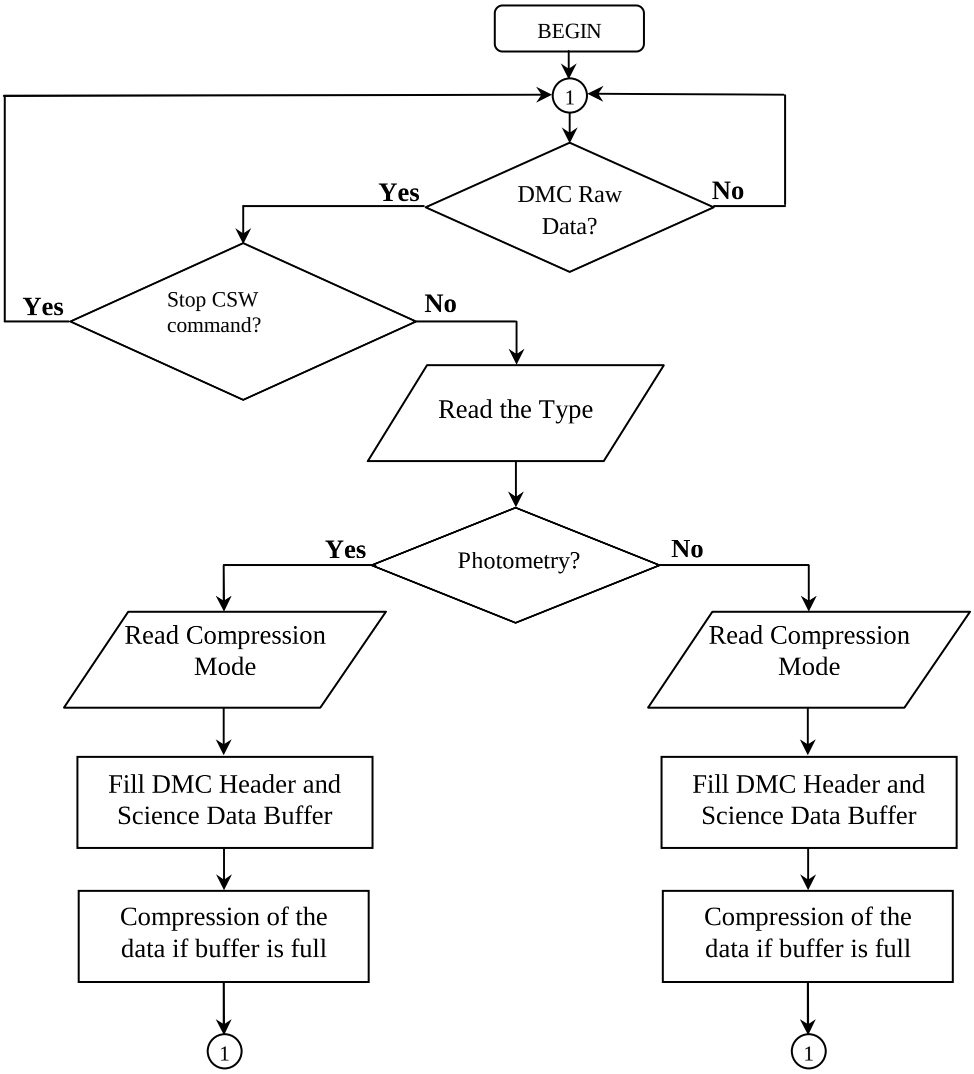


Figure 15. Flowchart of the Supervisor Algorithm

4.2.2.2 Compression Software

Figure 16 represents the general layout of the compression software modules. This part represents the core of the SPU HLSW. It is the SW part in which data is reduced and/or compressed using arithmetical and logical operations.

This module is responsible of:

- Science data compression following the DEC/MEC header logic
- DEC/MEC header compression
- Putting the compression results in the HK
- Building the compressed entity packet
- Transmission of the compressed data
- Raw channel data lossless compression

The CSW performs data reduction and compression according to the header information. It consists of a sequence of functions to fulfil the required compression mode.

4.2.2.2.1 HLSW Compression Modes

The SPU HLSW has 12 possible compression modes, which are listed and described below.

Compressed Data Transmission Mode

In the compressed data transmission mode, reduced and compressed data including DEC/MEC header have to be transmitted to DPU. They consist of either sample average rates in photometry mode or slope rates of ramps generated by the ramp-fitting module in spectroscopy imaging. Then, data may also be treated by the integration module to achieve the required compression ratio. This data will be further processed by redundancy reduction modules and finally by a lossless coding module, which will produce a stream of 16 bit words. The following compression modes are available:

- Default Compression Mode: **BOL1** (Photometry), **SPEC1** (Spectroscopy)
- Double Compression Mode: **BOL2** (Photometry), **SPEC2** (Spectroscopy)
- Half Compression Mode: **BOL0** (Photometry)
- Compressed Sensing Mode: BOLCS (Photometry)

Lossless Compression Mode

In the lossless compression mode, a selection of measurement values shall be transmitted without any loss of information, while the non-selected data are omitted. Therefore, data from selected detectors will be compressed (lossless compression only), and then transmitted to the DPU subsystem. The following modes are available:

- **BOL3** (Photometry), **SPEC3** (Spectroscopy)

Transparent Mode

In the transparent mode, data from selected detectors are transmitted without any compression. Furthermore, the DEC/MEC header is compressed and transmitted to DPU within the science data packet. The following modes are available:

- **BOL4** (Photometry), **SPEC4** (Spectroscopy).

Buffer Transmission Mode

In the buffer transmission mode, data from all detectors are transmitted without any compression. Therefore the whole data memory apart the output buffer will be filled with a continuous raw data stream. The following modes are available:

- **BUF_BOL** (Photometry), **BUF_SPEC** (Spectroscopy).

Compressed Sensing Mode (Photometry only)

In this compression mode a selection of samples from a projection onto an incoherent basis will be made. The following mode is available:

- **BOLCS** (Photometry).

Compression Mode for 4s Reset (Spectroscopy only)

In this compression mode a 4 seconds buffer will be subdivided into two 2-seconds buffers. Each buffer will be independently processed using the default compression mode in spectroscopy. The following mode is available:

- **SPEC5** (Spectroscopy).

4.2.2.2.2 Summary of Compression Modes

The compression modes are represented in *Table 4*. It shows the used functions for each mode.

HLSW Compression Mode	DXS	PRE	RBA/ RPF DEG	T_SRR	LLC	raw data sel.	Memory use	CMM
BOL0	•	•	•	•	•		1 Buffer	0x02
BOL1	•	•	•	•	•		1 Buffer	0x00
BOL2	•	•	•	•	•		1 Buffer	0x01
BOL3	•			•	•		1 Buffer	0x04
BOL4	•						1 Buffer	0x07
BUF_BOL							5 Buffers	0x09
BOLCS					•		1 Buffer	0x0C
SPEC1	•	•	•	•	•	•	1 Buffer	0x10
SPEC2	•	•	•	•	•	•	1 Buffer	0x11
SPEC3	•			•	•		1 Buffer	0x14
SPEC4	•						1 Buffer	0x17
SPEC5	•	•	•			•	2 Buffers	0x18
BUF_SPEC							5 Buffers	0x19

Table 4. Photometric and Spectroscopic compression modes

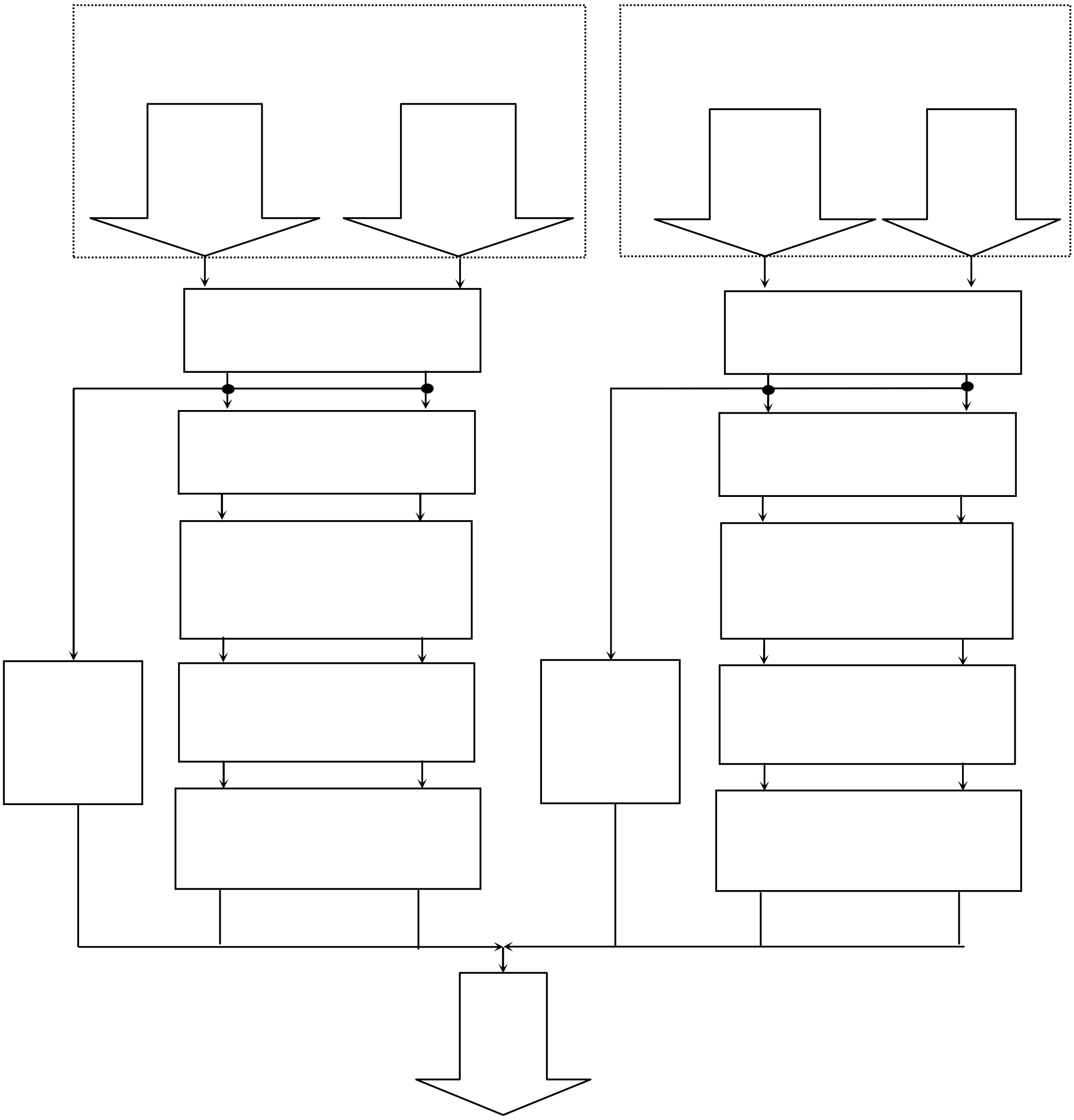


Figure 16. Compression SW Scheme.

4.2.3 The HK Module

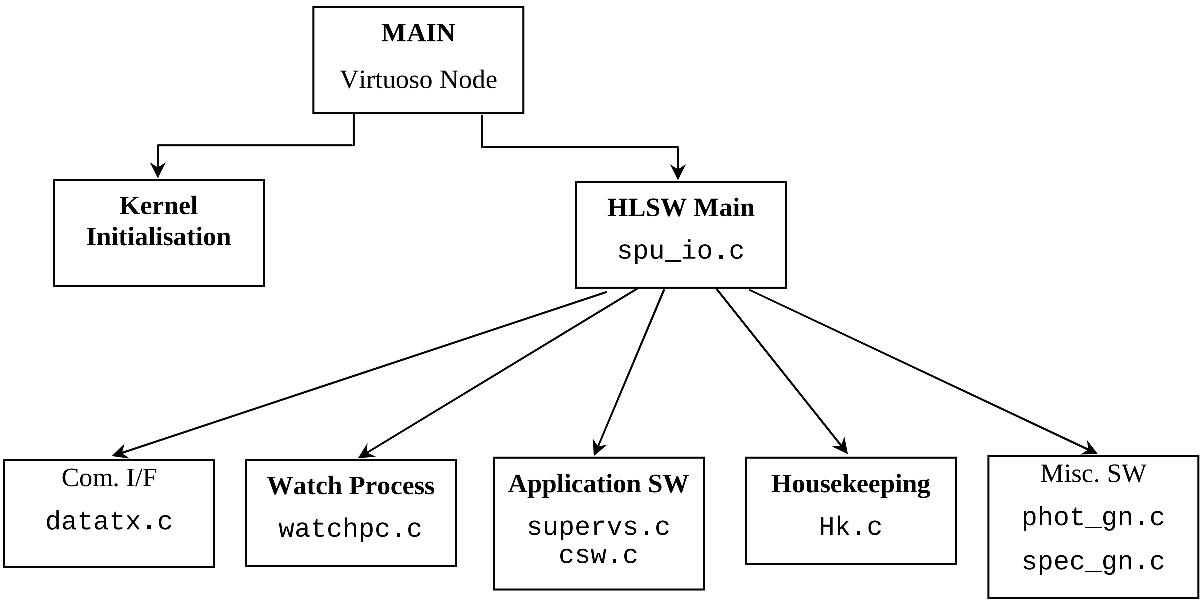
The SPU HLSW HK is sent from the SPU to the DPU. It consists either of compression results and status of the HLSW whenever raw data are received from DEC/MEC, or the SPU HLSW status while no observation is performed. Depending on their availability, the SPU HLSW HK are transmitted at a data rate compatible with the overall PACS HK.

The functionality of the SPU HLSW is continuously checked by the DPU monitoring some SPU HLSW HK data. The “Are you alive SPU” is described in the document RD007.

5 Component and Functional Description

SW Component	Files
Header Files	“bitmodel.h”, “pacscod.h”, “qsmodel.h”, “rangecod.h”, “genspu.h”, “spu_io.h”, “spuasw.h”, “spulib.h”
Low Level Driver Library	“l_dsp.h”, “l_errcod.h”, “l_gendef.h“, “l_hwmap.h“, “l_smcsco.h”, “l_smcsge.h”, “l_smcsin.h”, “l_smcsre.h”, “l_smcstr.h”, “l_itlmis.oba”, “l_memory.oba”, “l_memory.obc”, “l_smcsge.obc”, “l_smcsco.obc”, “l_smcsin.obc”, “l_smcsre.obc”, “l_smcstr.obc”, “l_dsp.oba”, “l_dsp.obc” , “l_pscgen.obc” , “l_eeprom.obc”
Communication Interface	“datatx.c”, ”dorada.s”, ”spu_io.c”, “supervs.c”, “watchpc.c”
Command Acknowledgment	“C2EEPROM.c”, “check.c”, “DMC_Con.c” “dump.c”, “Load.c”, “perform.c”, “Rc_Sel.c”, “spu_tst.c”, “Str_Stp.c”, “w_reset.c”, “write.c”
Compression Software	”average.c”, “bitmodel.c”, “bol_ex.c”, “bol1_2.c”, “bol3.c”, “bol4.c”, ”buf_bol.c”, ”buf_spec.c”, “csw.c”, “dmch_cp.c”, ”dxs.c”, “fill_in.c”, “fill_out.c” , “integ.c” , “llc.c” , “pacscod.c”, “p_proc.c” , “pacs_srt.c”, “qsmodel.c”, “ramp_ft.c”, “rangecod.c”, “spec1_2.c” , “spec3.c” , “spec4.c” , “T_S_Red.c”, “T_S_fm.c”
Housekeeping	“Hk.c”
Miscellaneous	“phot_gn.c”, “pkup_pg.c”, “spec_gn.c”, ”Spvs_Tst.c”, “rolib_a.s”

Table 5. Software Files of the SPU HLSW



5.1 Main Task

5.1.1 Type

`spu_io.c` – main task

5.1.2 Function

This is the main program task of the HLSW. It configures and initialises the connection SPU-DPU and SPU-DEC/MEC. When connection is established, it starts the HLSW task (Watch process, Supervisor, HK,…).

The following functions are contained:

`DpuConnect`.................................. Start the Link towards DPU
`ConnectLinkAsSlave`................ Start the Link Connection as Slave
`ConnectLinkAsMaster` .... Start the Link Connection as Master
`DMCLinkConnect` .... Start the Link Connection to DEC/MEC as Slave or Master
`LinkSmcsRead`.............................. Reads from SMCS communication link memory
`Link1355Write` .... Writes to SMCS communication link memory
`Memory_Error_Check`................ Checks for Single and/or Double Error Failure in memory
`MemoryScrubbing}`....................... Scrubs the Memory and correct Single Bit Error Failure in memory
`ResetLink` .... Resets the SMCS communication link
`SMCSHandlerC`.............................. Wrapper for Assembly ISR

5.1.2.1 SMCS Chip Handling by SPU HLSW

➢ **SPU Handover to HLSW**

The SPU HLSW is configured to start the DPU Link as Master (see link Start-up Protocol in RD005). When the SPU control is handed over from LLSW to the HLSW, the following steps are performed :

1. Reset SMCS chip
2. Set the DEC/MEC Link Connection Status in the HK to OFF
3. Set Nominal Configuration of the SMCS
4. Set DSP Interrupt at IRQ2 to Signal SMCS Events
5. Configure SMCS sub-interrupt for the two links (DPU and DMC). Unmask the following sub-interrupt (CH1_PAR_DIS_ERR, CH1_DATA_TXED, CH1_EOP_RXED, CH2_PAR_DIS_ERR, CH2_DATA_TXED and CH2_EOP_RXED)
6. Start the link to DPU as Master
7. Wait 9 seconds
8. If Connection to DPU is not established, then go back to Step 1.

9. Start the Application SW tasks
    a. Communication Tasks and HK
    b. Compression Tasks
    c. Tasks for SPU Test Mode

The link start-up to DMC is performed under request using the command CONNECT_DMC. The start-up protocol (Master/Slave) could be chosen by the user.

The DEC/MEC Link Connection Status in the HK is set to ON, whenever the connection between SPU and DMC is established.

- **Connection Loss between SPU and Other Subunits**
    - **DPU-SPU Disconnect Error**

When the SPU HLSW detects a disconnect error on the DPU link, the following steps are performed:

1. Stop the Application SW tasks
    i. Communication Tasks and HK
    ii. Compression Tasks
    iii. Tasks for SPU Test Mode
2. Perform the Steps 1-9 in the Paragraph 'SPU Handover to HLSW' above.

- **DMC-SPU Disconnect Error**

When the SPU HLSW detects a disconnect error on the DMC link, the DMC Link Status in the HK is set to OFF.

5.1.3 Interfaces

Call: KS_TaskGroupStart(SPU_INIT).....................Starts communication and HK tasks

Call: KS_TaskGroupStart(SPU_ASW)....................Starts compression software task

Call: KS_TaskGroupStart(SPU_TEST_TASKS) ....Starts data generators for photometry and spectr. tasks

Call: KS_TaskGroupAbort(SPU_INIT)...................Aborts communication and HK tasks

Call: KS_TaskGroupAbort(SPU_ASW) ....Aborts compression software task

Call: KS_TaskGroupAbort(SPU_TEST_TASKS)...Aborts data generators for photometry and spectr. Tasks

Call: KS_TaskSleep ....Time delay

5.1.4 Dependencies

None

5.1.5 Processing

See flowchart in Figure 17.

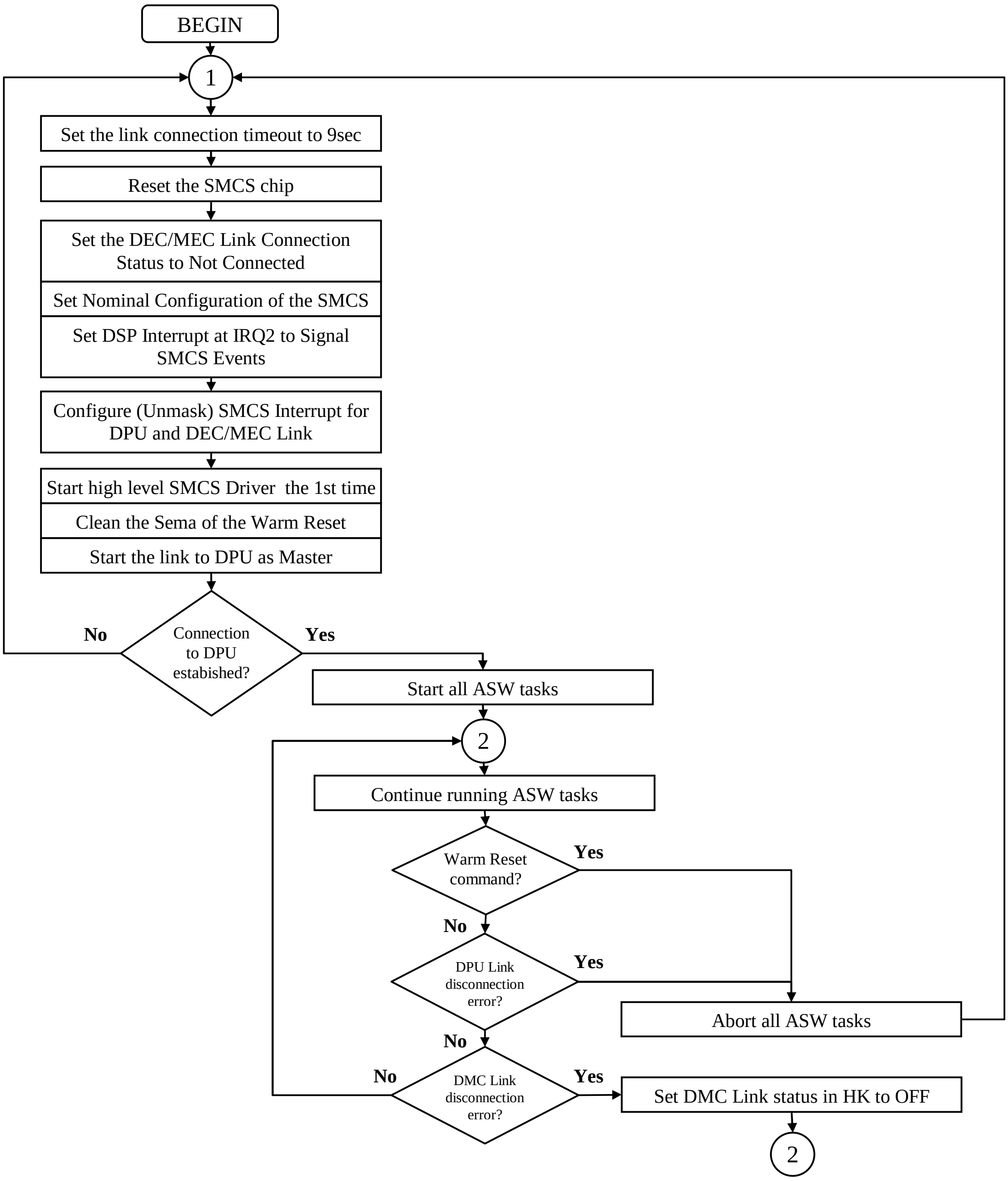


Figure 17. Flowchart of the Main Taks.

5.1.6 Data

None

5.1.7 Resources

SMCS Chip

5.2 Watch Process

5.2.1 Type

`watchpc.c` – task

5.2.2 Function

The watch process program listens to the DPU link. When a command is received, all running activities (except the HK task) are interrupted and the respective function is called to acknowledge and execute the command.

5.2.3 Interfaces

Call: Link1355Read ...... Reads from SMCS communication link towards DPU

Call: LinkSMCSRead .... Reads from SMCS communication link

Call: InitDetectorSelectionTables Initialise the Data Selection Table

Call: InitTables.............. Initialise the Bit Mask Table

Call: **KS_**FIFOPut......... Send the response packet to the FIFO (Data Transmission task)

Call: KS_SemaSignal .... Signals the start of HK transmission

Call: KS_TaskSleep ...... Time delay

Call: Load...................... Calls procedure to acknowledge and perform the Load Command request

Call: Dump .... Calls procedure to acknowledge and perform the Dump Command request

Call: Check.................... Calls procedure to acknowledge and perform the Check Command request

Call: Perform................. Calls procedure to acknowledge and perform the Perform Activity command request

Call: Write..................... Calls procedure to acknowledge and perform the Write Command request

5.2.4 Dependencies

Called from the main task.

5.2.5 Processing

See flowchart in Figure 18.

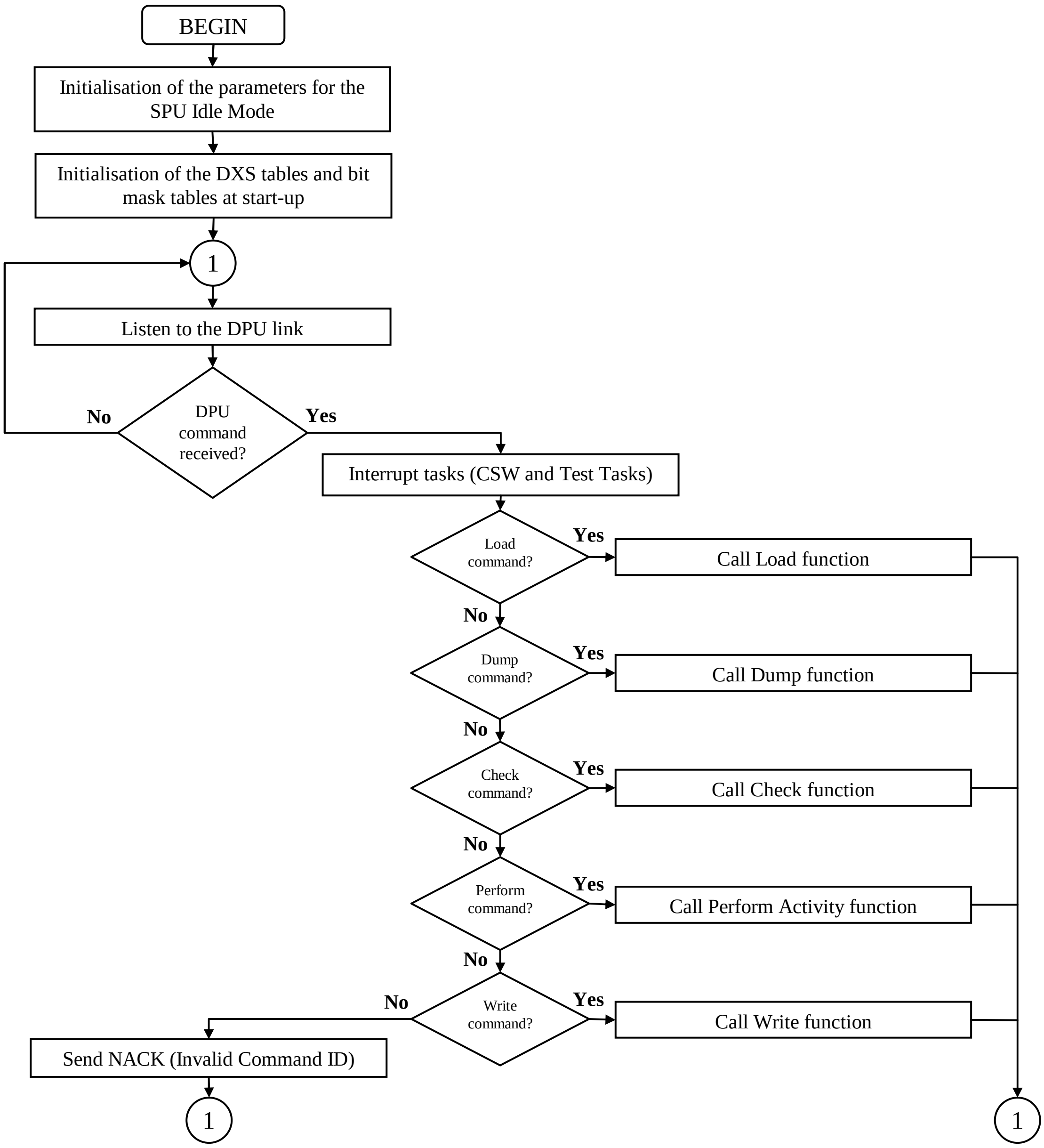


Figure 18. Flowchart of the Watch Process Task

5.2.6 Data

Read data from SMCS Link1.

5.2.7 Resources

SMCS Chip

5.3 Supervisor

5.3.1 Type

`supervs.c` – task

5.3.2 Function

The supervisor task listens to the DMC link. It reads DMC packets whenever received. If a start CSW command is signalled, the DMC packets are buffered and the compression is started. Otherwise, new DMC packets will overwrite the old ones (DMC data are ignored).

Furthermore, the Supervisor prepares the lookup tables from the received DMC header and makes them ready for the CSW, whenever the CSW is started.

The DMC packet structure is outlined below. More details can be found in RD009.

DEC/MEC Raw Data Stream to SPU

Figure 19 represents one detector array in spectroscopy. They consist of a total of 18x25 detectors. Each detector will deliver a 16-bit signal. One row of 18 detector data is added at the DEC level. Therefore, the total science data frame size is 468 words.

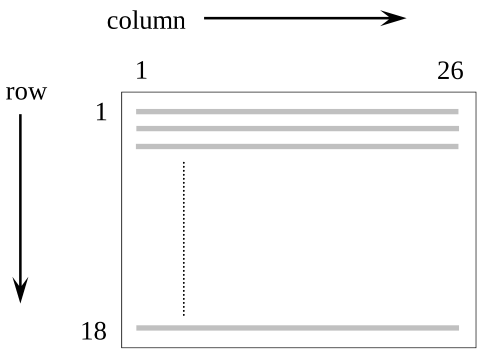


Figure 19. Detector array for the Spectroscopy.

*Figure 20*a represents the 8 arrays of the SWL photometer and *Figure 20*b the 2 arrays of the LWL photometer. They consist of a total of 2048 detectors for SWL and 512 detectors for LWL. Each detector will deliver a 16-bit signal.

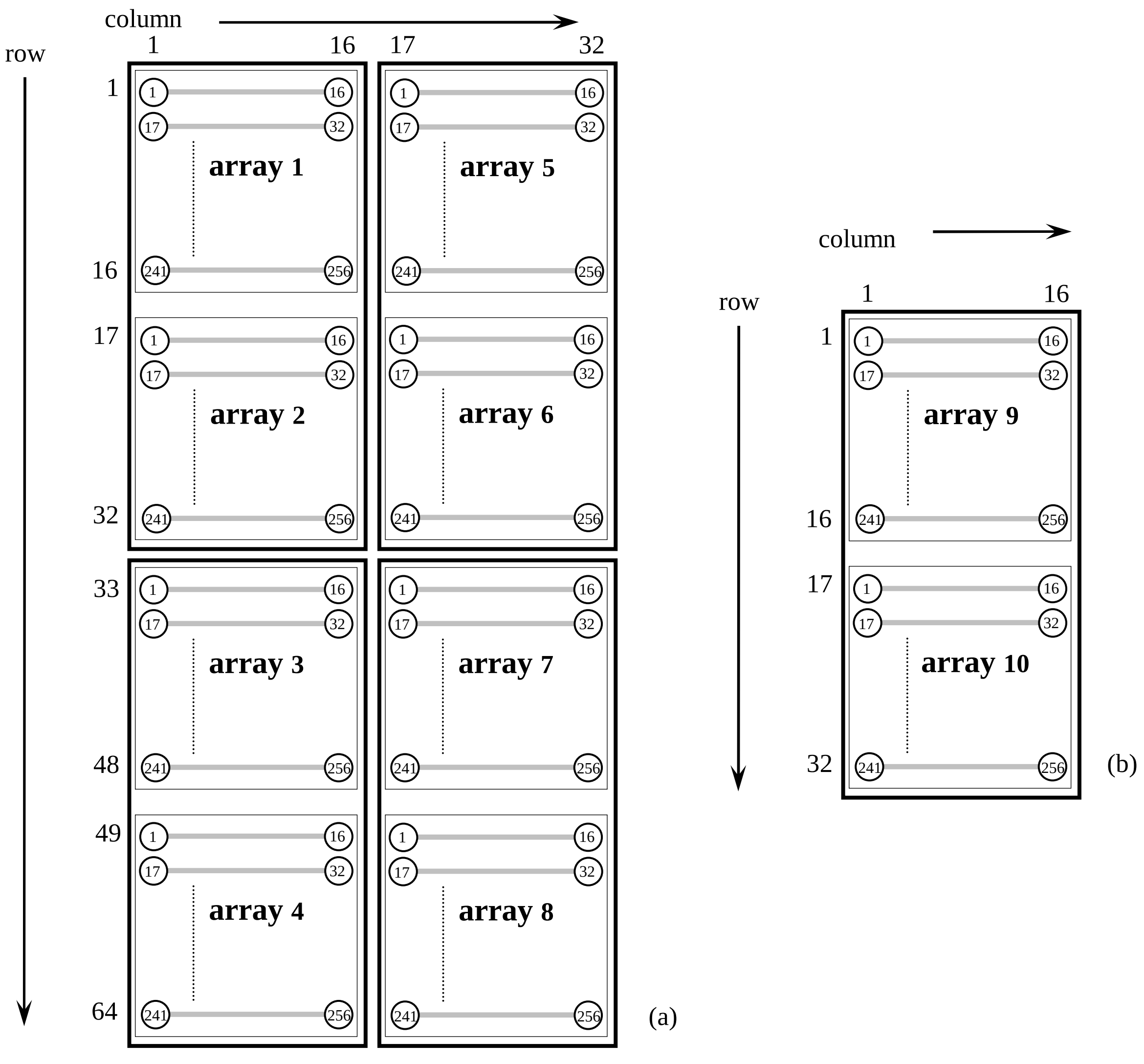


Figure 20. Configuration of the (a) SWL-Detector Arrays and of the (b) LWL-Detector Arrays.

The maximum packet size sent from DEC/MEC to SPU is 250 words (64 Bytes DMC header + 26x18x16/8 Bytes science data = 1000 Bytes) in spectroscopy and 272 words (64 Bytes DMC header + 16x16x2x16/8 Bytes science Data = 1088 Bytes) in photometry as depicted in *Figure 21* and *Figure 22*.

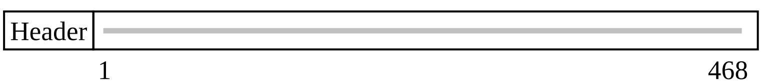


Figure 21. Resulting DMC packet structure for Spectroscopy.

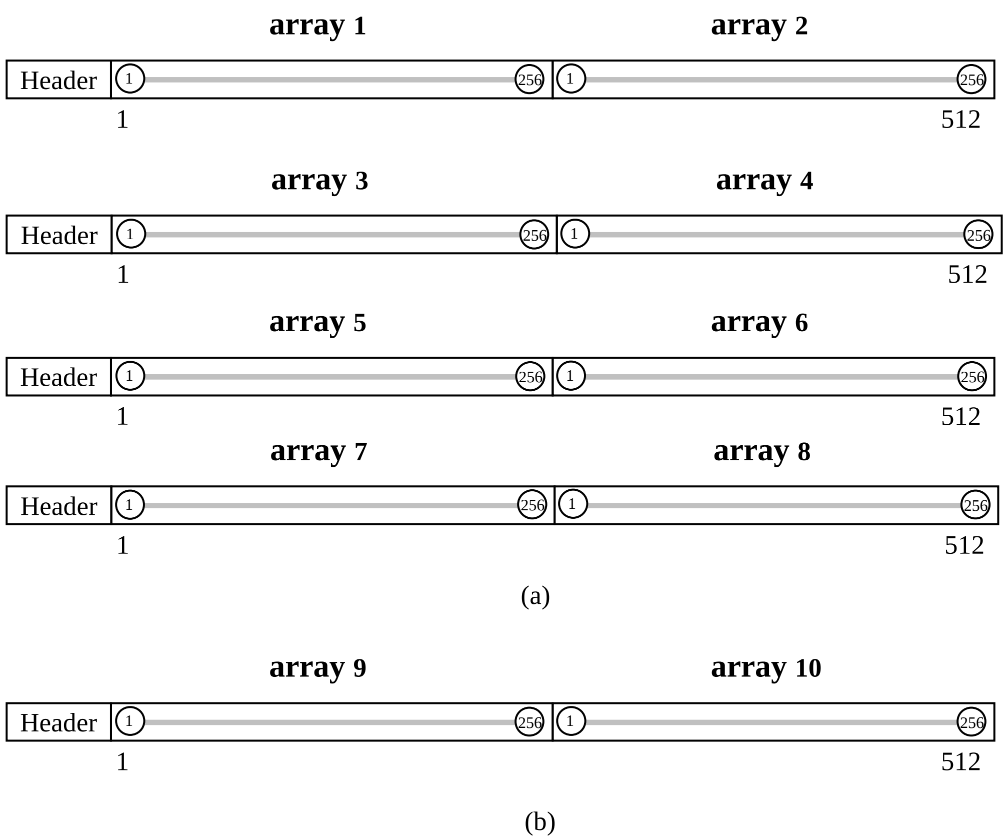


Figure 22. Resulting DMC packet structure for (a) SWL-Photometry and for (b) LWL-Photometry.

DEC/MEC Header Structure

The DEC/MEC header structure is represented in RD016 (HLSW User Manual)

Label (LBL)

The Label statement is meant to indicate to the SPU-HLSW across which "quadruples" it has to average in photometry. It is further used also to indicate the number of discrete chopper steps within a sequence. In spectroscopy the Label statement is used in wavelength switching mode when Bit 1 and Bit 6 are set. Hereafter, the following conventions for the Label in photometry:

If Label is set to 0, then no sequence is active otherwise

Bit 1 … is set to 1, to indicate that DMC sequence is active.

Bit 2-6 … counts chopper positions within sequence
Bit 7 … signal from BB1
Bit 8 … signal from BB2

LABEL = 63 has a special meaning and indicates "freeze frame mode"

Validity (VLD)

The “Validity” statement defines if the data is valid or not. If the data is declared invalid the data will be ignored and the buffer is cleaned until valid data is arriving.

Compression Parameters

The compression and transmission mode which corresponds to the Compression Parameters field in are listed in Table 6.

Comp. and Trans. Mode ID	**Photometry**	**Spectroscopy**
0x00	Default Mode	None
0x01	Double Compression Mode	None
0x02	Half Compression Mode	None
0x04	Lossless Compression Mode	None
0x07	Transparent Mode	None
0x09	Buffer Transmission Mode	None
0x10	None	Default Mode
0x11	None	Double Compression Mode
0x14	None	Lossless Compression Mode
0x17	None	Transparent Mode
0x18	None	Compression Mode for 4 second reset
0x19	None	Buffer Transmission Mode

Table 6. List of Compression Parameters for Photometry and Spectroscopy

5.3.3 Interfaces

Call: Link1355Read ....Reads from SMCS link towards DPU

Call: LinkSmcsRead................................Reads from SMCS link

Call: WriteInCircularBuffer ....Calls procedure to fill the Input Buffer with Science data

Call: KS_SemaSignal(SEMA_ASW)......Signals the start of the compression software task

Call: ResetCircularBuffers ....Memory allocation for buffering of DMC header, science and telemetry data

5.3.4 Dependencies

Called from the main task.

5.3.5 Processing

See flowchart in Figure 23, Figure 24, Figure 25 and .

‘`Compress_Now = true`’ in the flowchart means: “Close the actual buffer and make it ready for compression. Begin to store data in a new buffer.”

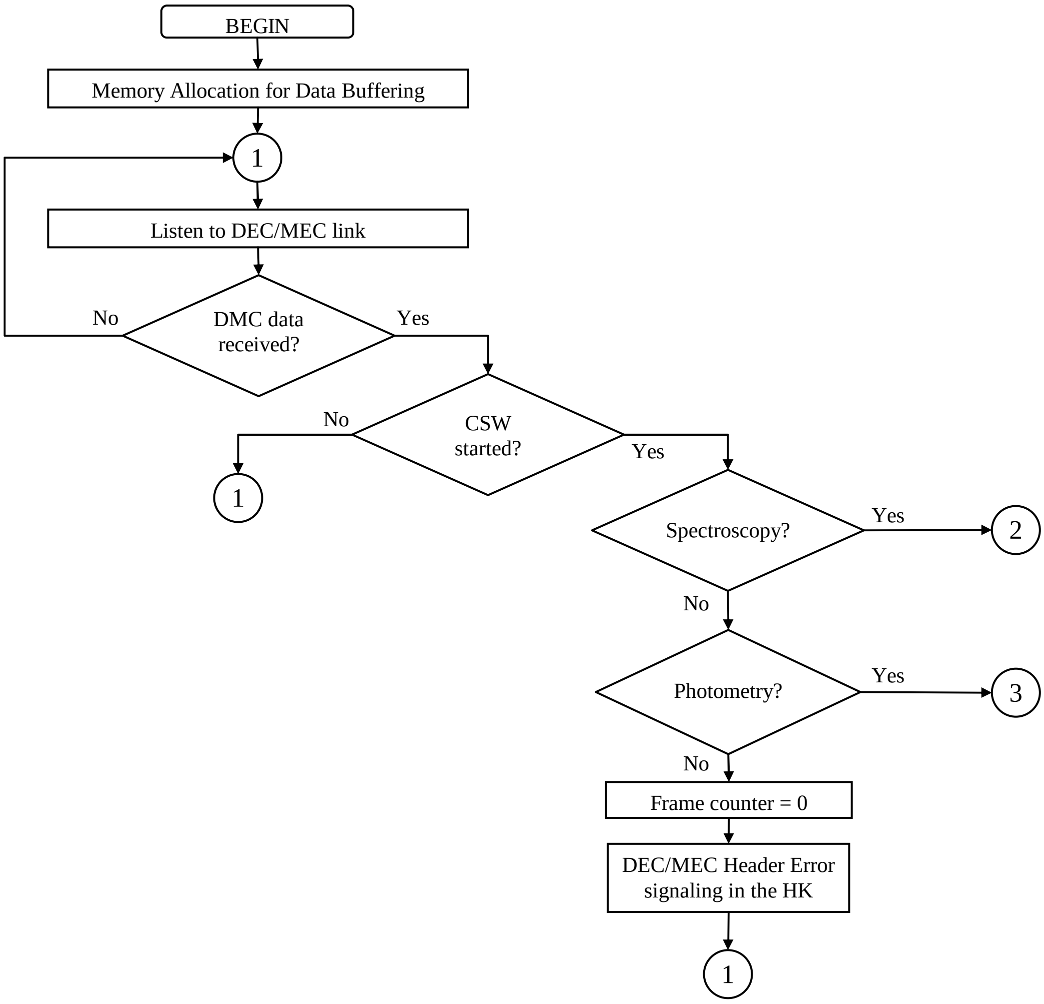
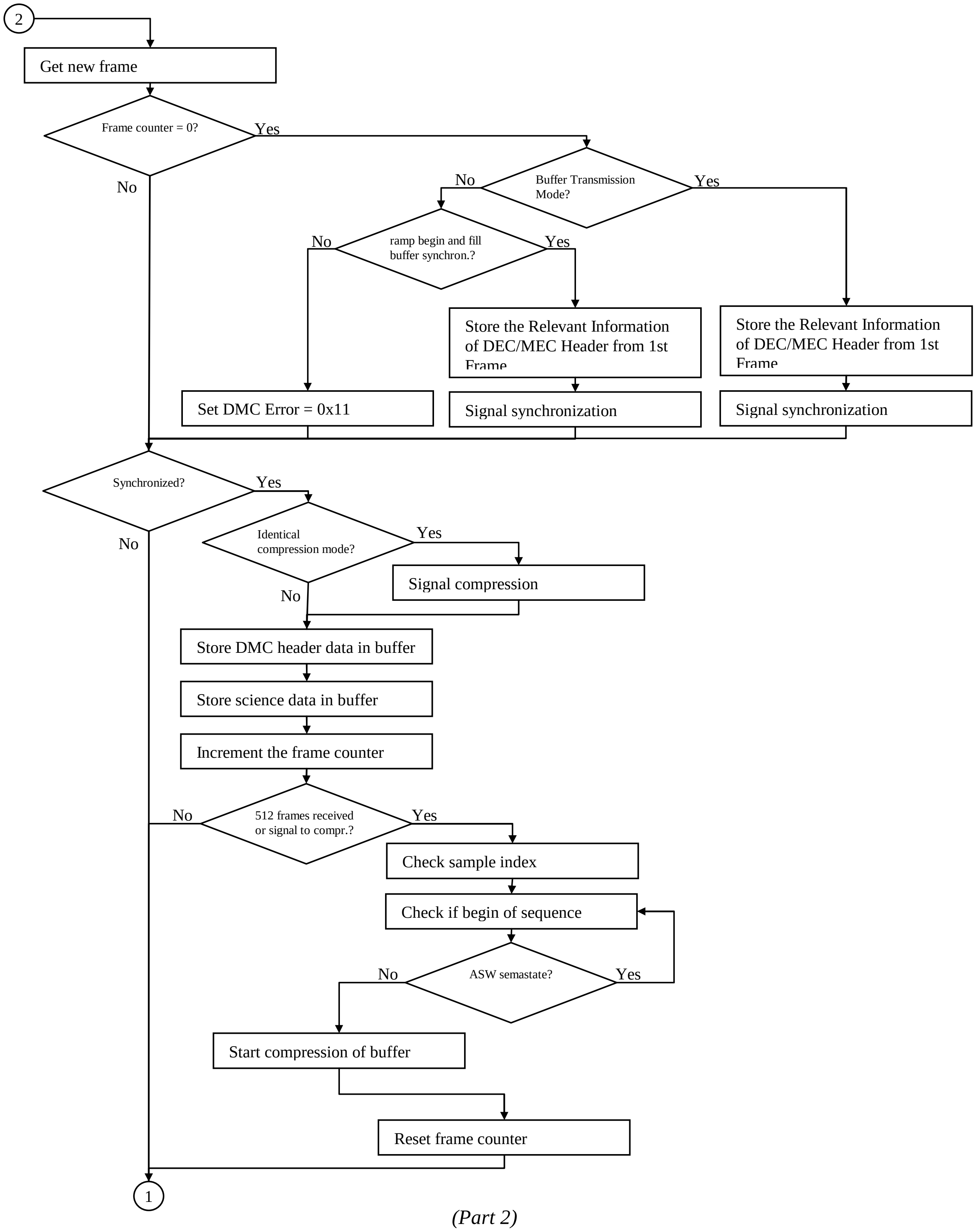


Figure 23. Flowchart of the Supervisor Task (Part 1)

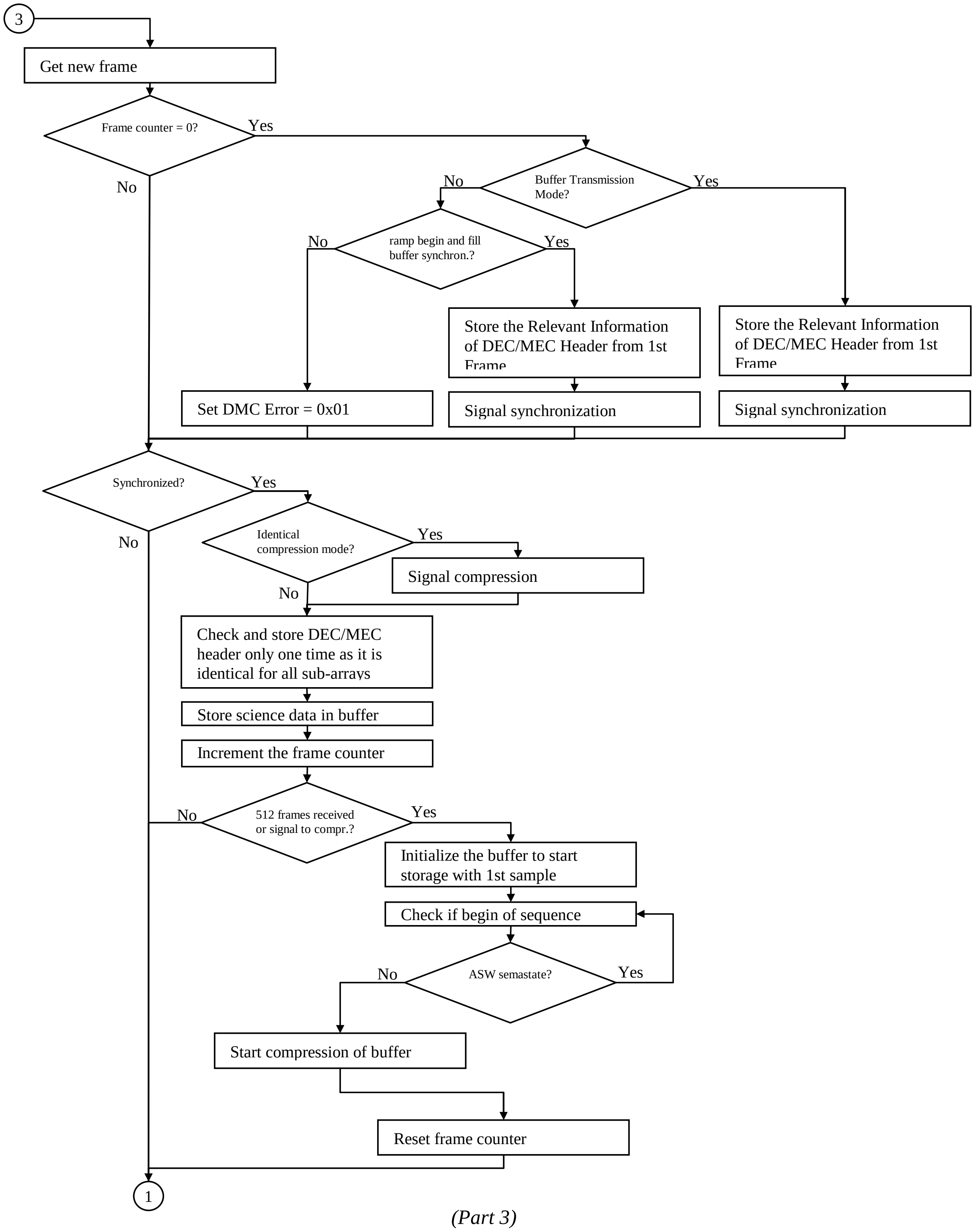
Figure 24. Flowchart of the Supervisor Task (Part 2)
see next page

Figure 25. Flowchart of the Supervisor Task (Part 3)
see page after next



2
Get new frame
Frame counter = 0?
Yes
No
Buffer Transmission Mode?
No
Yes
ramp begin and fill buffer synchron.?
No
Yes
Store the Relevant Information of DEC/MEC Header from 1st Frame
Store the Relevant Information of DEC/MEC Header from 1st Frame
Set DMC Error = 0x11
Signal synchronization
Signal synchronization
Synchronized?
Yes
No
Identical compression mode?
Yes
No
Signal compression
Store DMC header data in buffer
Store science data in buffer
Increment the frame counter
512 frames received or signal to compr.?
No
Yes
Check sample index
Check if begin of sequence
ASW semastate?
No
Yes
Start compression of buffer
Reset frame counter
1

(Part 2)



3
Get new frame
Frame counter = 0?
Yes
No
Buffer Transmission Mode?
No
Yes
ramp begin and fill buffer synchron.?
No
Yes
Store the Relevant Information of DEC/MEC Header from 1st Frame
Store the Relevant Information of DEC/MEC Header from 1st Frame
Set DMC Error = 0x01
Signal synchronization
Signal synchronization
Synchronized?
Yes
No
Identical compression mode?
Yes
Signal compression
No
Check and store DEC/MEC header only one time as it is identical for all sub-arrays
Store science data in buffer
Increment the frame counter
512 frames received or signal to compr.?
No
Yes
Initialize the buffer to start storage with 1st sample
Check if begin of sequence
ASW semastate?
No
Yes
Start compression of buffer
Reset frame counter
1

(Part 3)

5.3.6 Data

Read data from SMCS Link2.

5.3.7 Resources

SMCS Chip

1 MB allocated Data Memory

5.4 Data Transmission

5.4.1 Type

`datatx.c` – task

5.4.2 Function

The data transmission program module waits for the tasks outputs and sends them to the DPU. This task organizes the data flow from SPU to DPU.

5.4.3 Interfaces

Call: LinkSmcsWrite ....Writes to SMCS communication link

Call: KS_FIFOGetW ....Wait that another task writes in the FIFO

5.4.4 Dependencies

Called from the main task.

5.4.5 Processing

See flowchart in Figure 26.

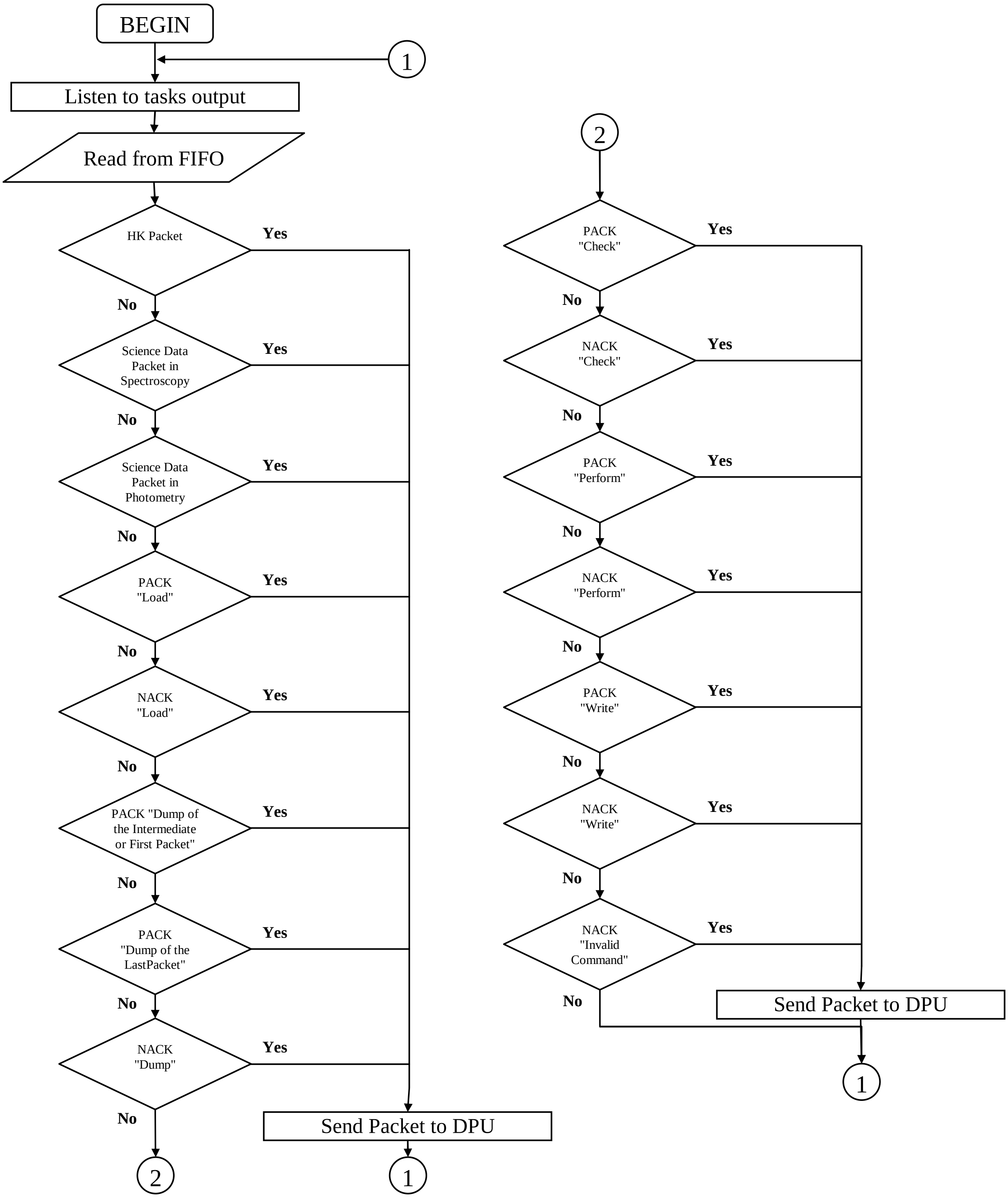


Figure 26. Flowchart of the Data Transmission Task.

5.4.6 Data

Data read from FIFO.

5.4.7 Resources

SMCS Chip

5.5 Housekeeping

5.5.1 Type

`hk.c` – task

5.5.2 Function

This program module produces HK parameters at 1.9 seconds interval. One relevant parameter is an incremental counter as described in RD007 (“Are You Alive SPU”). When the CSW is started some parameters are read from HK buffer (compression information).

Following parameters are included in the HK:

HK_HEADER .... Housekeeping Header
OBSID .... Observation Identification
PIX .... Packet Index
CI .... Counter Increment
REAL .... Reduction Algorithm used
SATUR_FLAG .... Saturation Flag
SAMP_CORR .... Glitch counter information
MAINT_RAMPS .... Number of maintained subramps
CPU_WORKLOAD .... CPU workload
DMC_LINK_STATUS .... DEC/MEC link status
INTEG_RAMPS .... Number of integrated ramps
VID .... Version Identification
RCX .... Raw Channel Index
DMC_ERROR .... DEC/MEC header error
MEM_STATUS_CNTS ...... Counters for EDAC Memory Checks
SUBVERSION .... Subversion of the SPU HLSW
LLC_ERROR .... Invalid LLC Parameter
PAR_MONITOR .... Monitored Write Parameter

5.5.3 Interfaces

Call: KS_FIFOPut ....Sends the HK packet to the FIFO (Data Transmission task)

Call: KS_TaskSleep ....Time delay

5.5.4 Dependencies

Called from the main task.

5.5.5 Processing

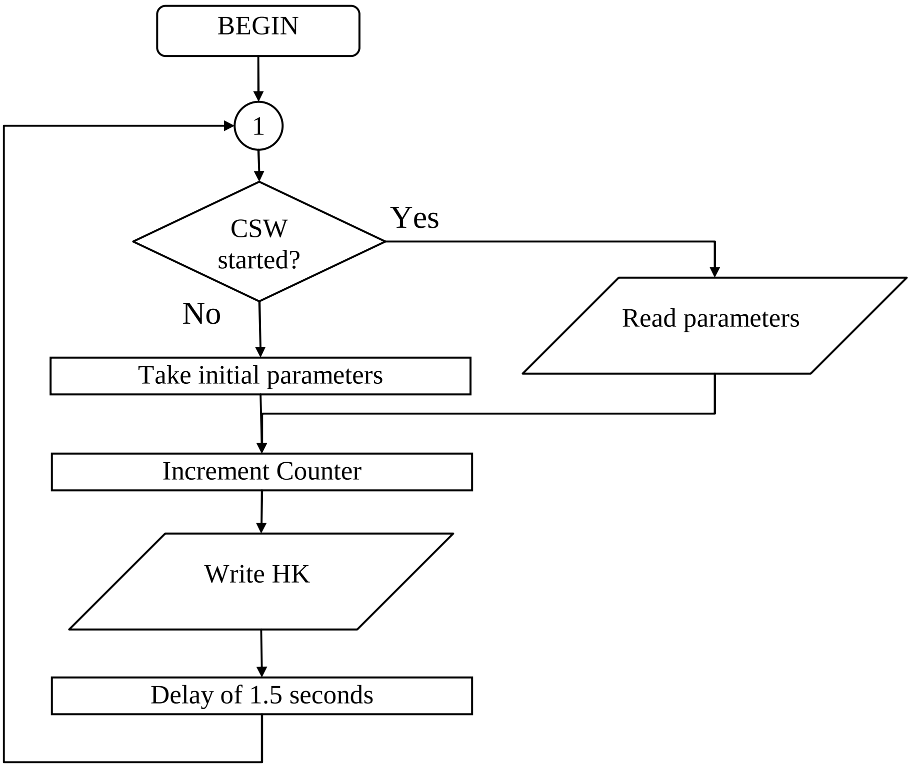


Figure 27. Flowchart of the Housekeeping Task

5.5.6 Data

Writes data to the FIFO.

5.5.7 Resources

None

5.6 Fill_InputBuffer

5.6.1 Type

`fill_in.c` – procedure

5.6.2 Function

The Fill_InputBuffer programme module is used to handle the data flow from/to the memory. It consists of circular buffer and functions to load and dump data to/from the memory.

Contains of following functions:

`DeclareCircularBuffer` .... Reserves a memory space in the SPU RAM for data buffering

`WriteInCircularBuffer` .... Fills the input buffer with the DEC/MEC science data

`ReadInCircularBuffer` .... Reads the science data from input buffer

`CircularBufferQuerySpace` .. Checks the empty memory space in science data input buffer

`CircularBufferQueryUsed` ..... Checks the used memory space in science data input buffer

`CircularBufferRemoveLastWrittenValues..` Remove the last values from a circular buffer

`CircularBufferRemove` .... Clears a circular buffer by depletion

`ResetCircularBuffers` .... Memory allocation for buffering of science header, DMC and telemetry data

5.6.3 Interfaces

None

5.6.4 Dependencies

Called from Supervisor or Supervisor_Test.

5.6.5 Processing

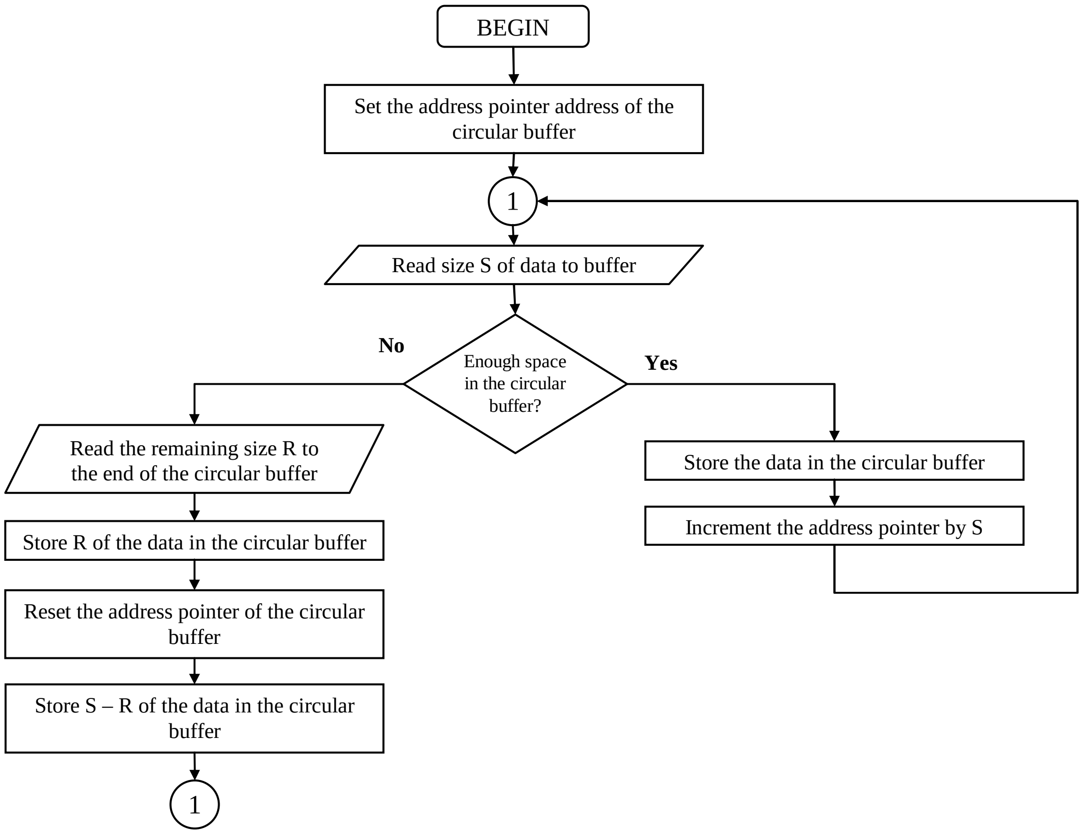


Figure 28. Flowchart of the Fill_InputBuffer Procedure

5.6.6 Data

None

5.6.7 Resources

2 MB Memory

5.7 Command Interpretation and Acknowledgement

5.7.1 LOAD Command

5.7.1.1 Type

`load.c` – procedure

5.7.1.2 Function

This procedure acknowledges and executes (if Load parameters are correct) the Load command received from DPU. This command is used to transfer data in SPU DRAM and DPRAM. For more details see RD007.

5.7.1.3 Interfaces

Call: KS_FIFOPut ....Sends the response packet to the FIFO (Data Transmission task)

5.7.1.4 Dependencies

Called from watch process.

5.7.1.5 Processing

See flowchart in Figure 29.

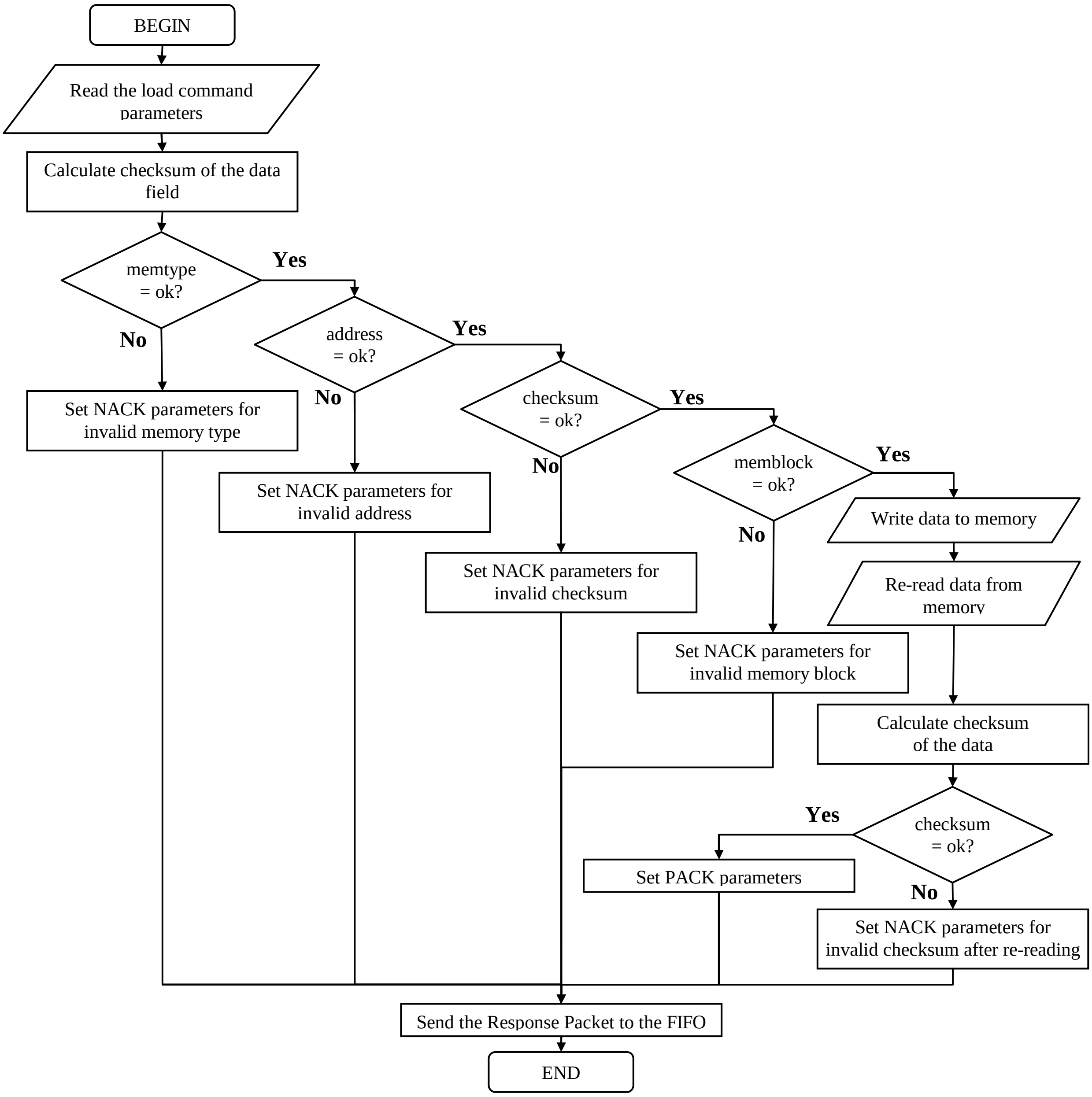


Figure 29. Flowchart of the Procedure for Load Command Handling

5.7.1.6 Data

Writes data to the FIFO.

5.7.1.7 Resources

None

5.7.2 DUMP Command

5.7.2.1 Type

`dump.c` – procedure

5.7.2.2 Function

This procedure acknowledges and executes (if Dump parameters are correct) the Dump command received from DPU. This command is used to transfer data from SPU RAM. For more details see RD007.

5.7.2.3 Interfaces

Call: KS_FIFOPut..................Sends the response packet to the FIFO (Data Transmission task)

Call: KS_TaskSleep ....Time delay

5.7.2.4 Dependencies

Called from watch process.

5.7.2.5 Processing

See flowchart in *Figure 30*.

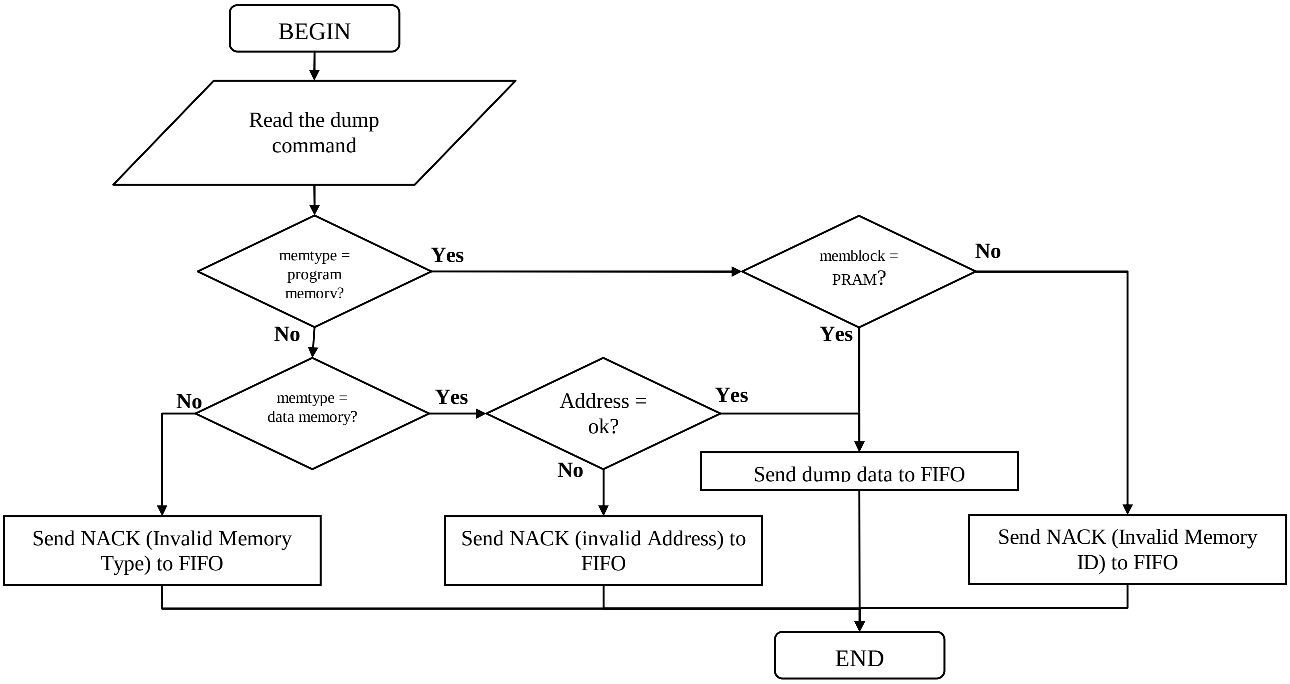


Figure 30. Flowchart of the Procedure for Dump Command Handling

5.7.2.6 Data

Writes data to the FIFO.

5.7.2.7 Resources

None

5.7.3 CHECK Command

5.7.3.1 Type

`check.c` – procedure

5.7.3.2 Function

This procedure acknowledges and executes (if Check parameters are correct) the Check command received from DPU. This command is used to check the SPU RAM. For more details see RD007.

Contains of following functions:

`Crc` .... Calculates the Checksum

5.7.3.3 Interfaces

Call: KS_FIFOPut .... Sends the response packet to the FIFO (Data Transmission task)

5.7.3.4 Dependencies

Called from watch process.

5.7.3.5 Processing

See flowchart in Figure 31

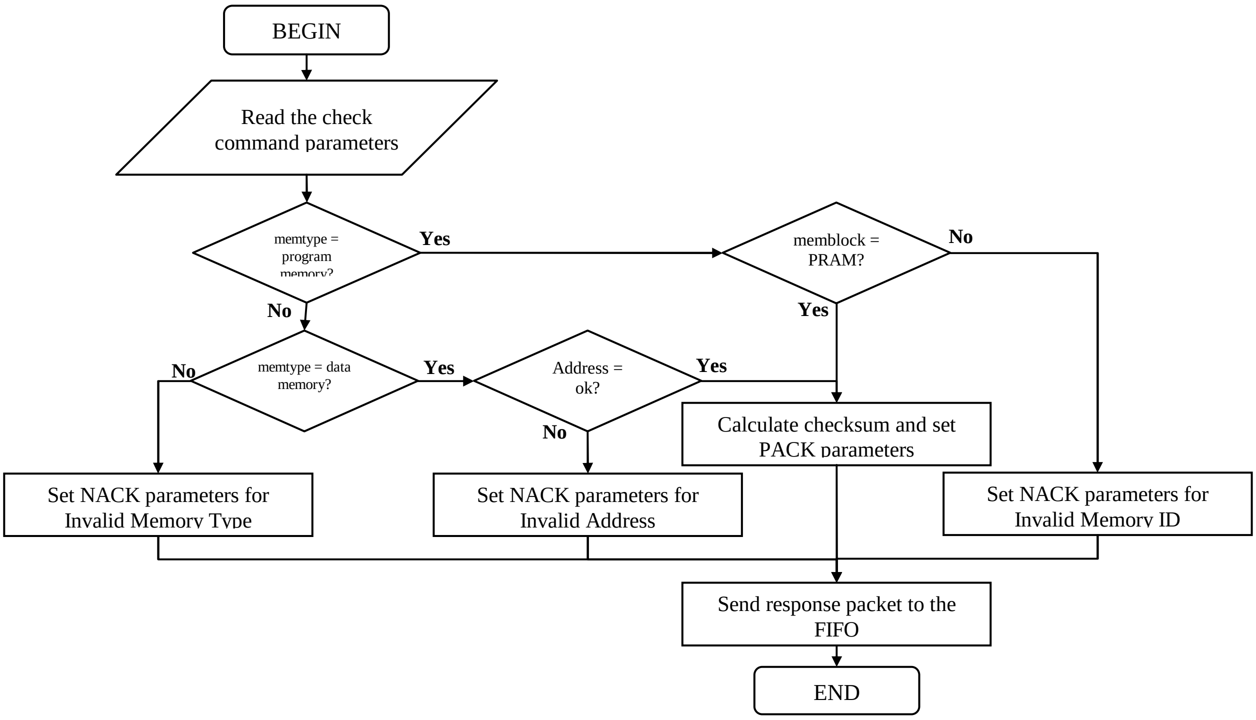


Figure 31. Flowchart of the Procedure for Check Command Handling

5.7.3.6 Data

Writes data to the FIFO.

5.7.3.7 Resources

2 MB Data Memory

5.7.4 Perform Activity Command

5.7.4.1 Type

`perform.c` – procedure

5.7.4.2 Function

This procedure acknowledges the reception of the Perform command and starts the appropriate function according to the received Activity ID.

5.7.4.3 Interfaces

Call: PerformCopyToEeprom ....Copy Data from RAM to EEPROM

Call: PerformWarmReset ....Warm Reset function

Call: PerformRawChannelSelection .....Raw Channel Selection function

Call: PerformStartStop ....Start or Stop CSW function

Call: PerformSpuTestMode..................SPU Test for Spectroscopy or Photometry function

Call: PerformConnectToDmc...............Start Link to DEC/MEC as Master or Slave

Call: KS_FIFOPut ....Sends the response packet to the FIFO (Data Transmission task)

5.7.4.4 Dependencies

Called from watch process.

5.7.4.5 Processing

See Flowchart in *Figure 32*.

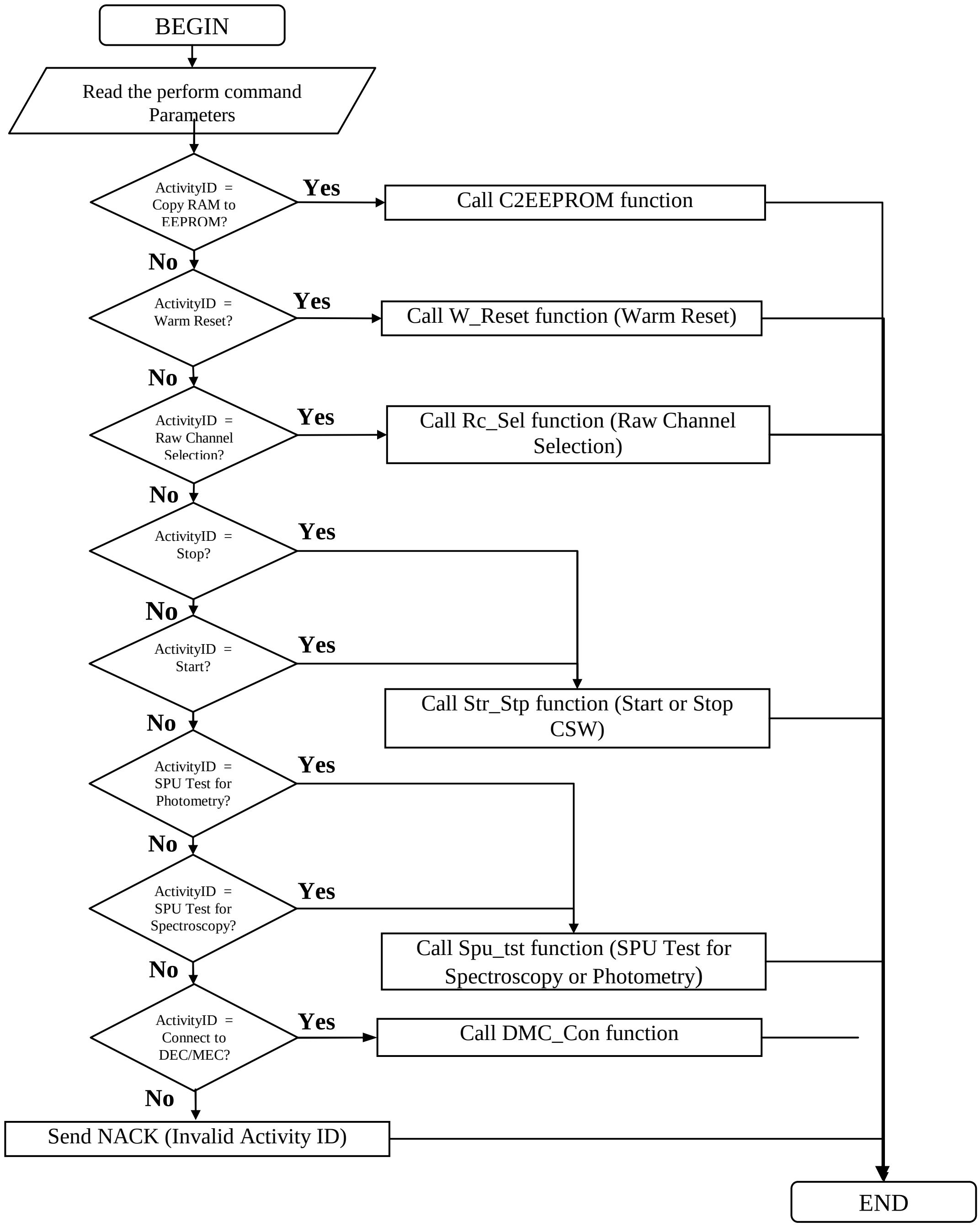


Figure 32. Flowchart of the Procedure for the Perform Activity Command Handling

5.7.4.6 Data

Writes data to the FIFO.

5.7.4.7 Resources

None

5.7.5 CP_DATA_RAM_EEPROM Command

5.7.5.1 Type

`C2EEPROM.c` – procedure

5.7.5.2 Function

This procedure acknowledges the reception of the Copy data from RAM to EEPROM command and updates the HLSW in SPU EEPROM. It Copies the HLSW from PRAM and/or DRAM to EEPROM..

5.7.5.3 Interfaces

Call: PerformCopyToEeprom Copies Data from RAM to EEPROM

Call: KS_FIFOPut..................Sends the response packet to the FIFO (Data Transmission task)

Call: KS_TaskSleep ....Time delay

5.7.5.4 Dependencies

Called from perform activity command module.

5.7.5.5 Processing

See flowchart in Figure 33.

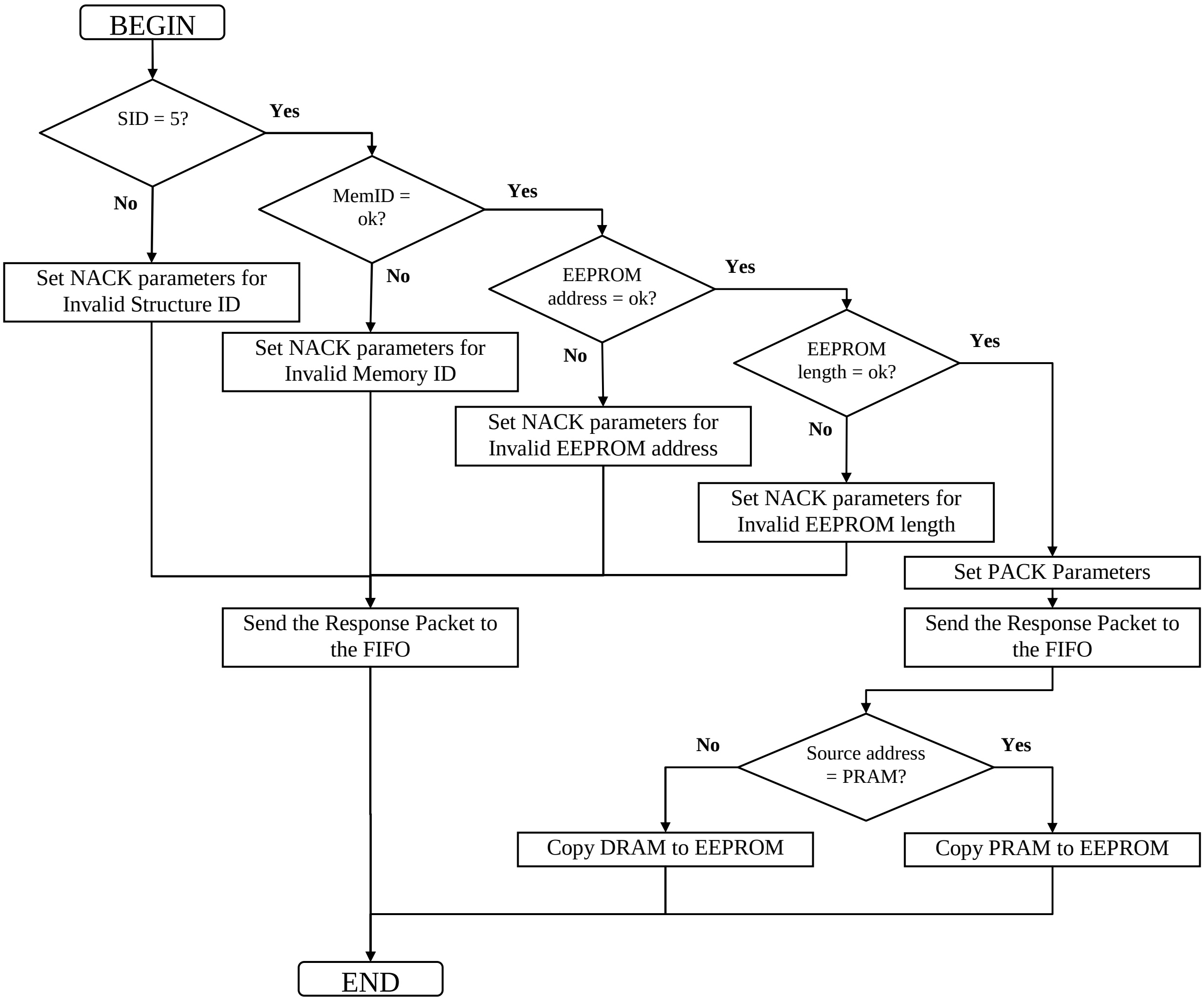


Figure 33. Flowchart of the Procedure for Copy Data from RAM to EEPROM Command Handling

5.7.5.6 Data

Writes data to the FIFO.

5.7.5.7 Resources

None

5.7.6 RESET Command

5.7.6.1 Type

`w_reset.c` – procedure

5.7.6.2 Function

This procedure performs a Warm Reset of the HLSW. If SPU HLSW receives the Warm Reset command it cleans all buffers (DCMH data, science data and telemetry data buffers), reset the HK counter CI and PIX, kill all tasks (Supervisor, Watch Process, Data Transmission and HK) , reset the SMCS chip and restart the link connection to DPU.

5.7.6.3 Interfaces

Call: KS_FIFOPut ....Sends the response packet to the FIFO (Data Transmission task)

Call: KS_TaskSleep ....Time delay

Call: KS_SemaSignal(WARM_RESET_SEMA).......... Reset SMCS chip and all the tasks apart the spu_io

5.7.6.4 Dependencies

Called from perform activity command module.

5.7.6.5 Processing

See flowchart in Figure 34.

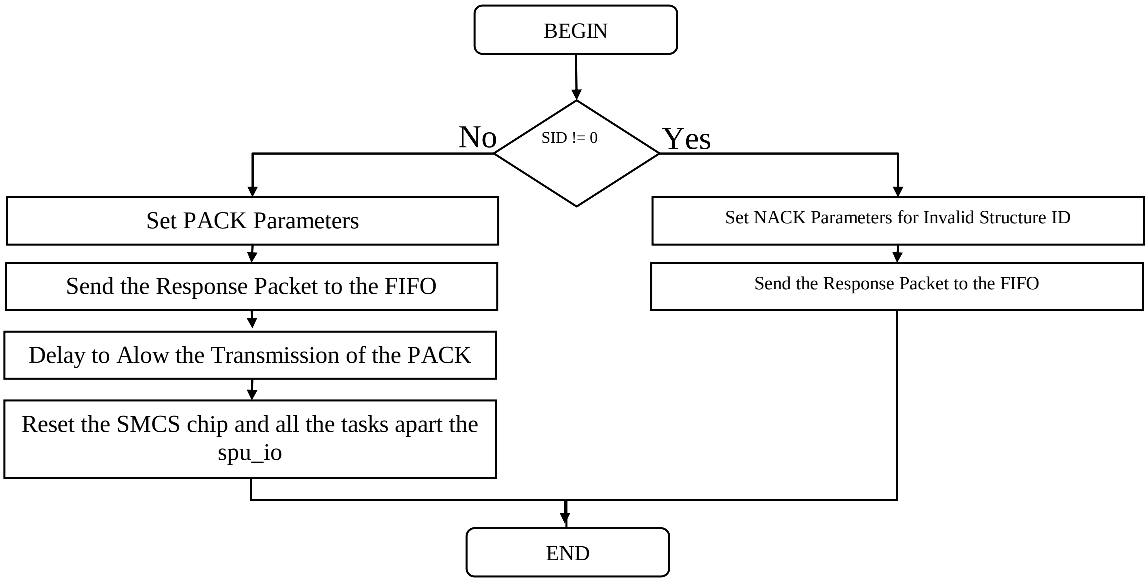


Figure 34. Flowchart of the Procedure for Warm Reset Command Handling

5.7.6.6 Data

Writes data to the FIFO.

5.7.6.7 Resources

None

5.7.7 RAW_CHAN_TRAN_MODE Command

5.7.7.1 Type

`Rc_Sel.c` – procedure

5.7.7.2 Function

This procedure is used to select the number and index of channels from which data should be lossless compressed and transmitted. The channels index and number could be selected for photoconductors and bolometers. At startup this index and number are set to zero.

5.7.7.3 Interfaces

Call: KS_FIFOPut .... Sends the response packet to the FIFO (Data Transmission task)

5.7.7.4 Dependencies

Called from perform activity command module.

5.7.7.5 Processing

See flowchart in Figure 35.

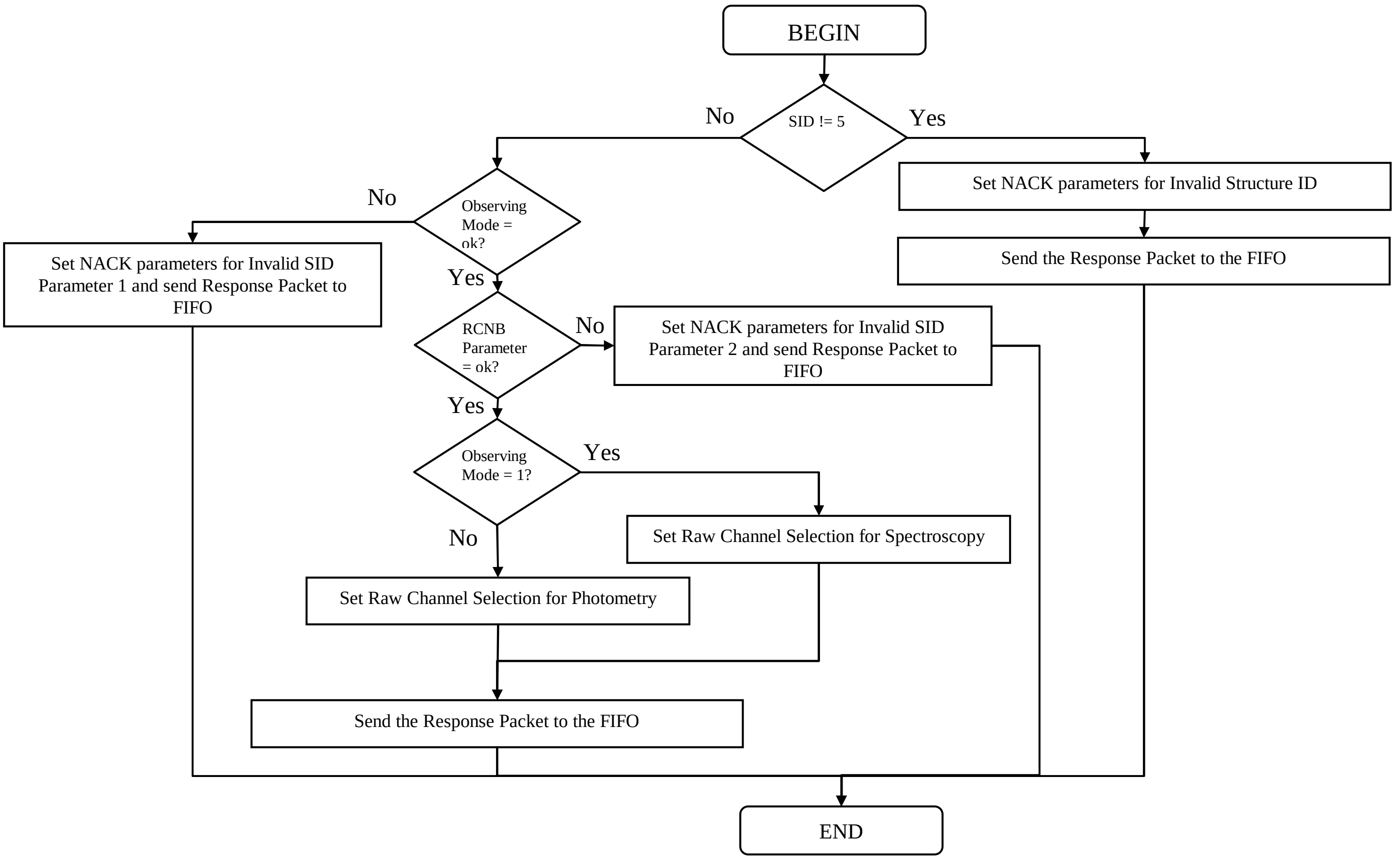


Figure 35. Flowchart of the Procedure for Raw Channel Selection Command Handling

5.7.7.6 Data

Writes data to the FIFO.

5.7.7.7 Resources

None

5.7.8 START/STOP_REDUCT_COMPR Command

5.7.8.1 Type

`Str_Stp.c` – procedure

5.7.8.2 Function

This procedure is used to Start or Stop the CSW.

- The **Start** command is performed to run the SPU CSW for DEC/MEC data.
- The **Stop** command is used to abort any running activity. When called it deletes the buffered data (DCMH data, science data and telemetry data buffers).

5.7.8.3 Interfaces

Call: KS_FIFOPut .... Sends the response packet to the FIFO (Data Transmission task)

Call: KS_TaskSleep .... Time delay

Call: ResetCircularBuffers .... Memory allocation for buffering of science header, DMC and telemetry data

5.7.8.4 Dependencies

Called from perform activity command module.

5.7.8.5 Processing

See flowchart in Figure 36.

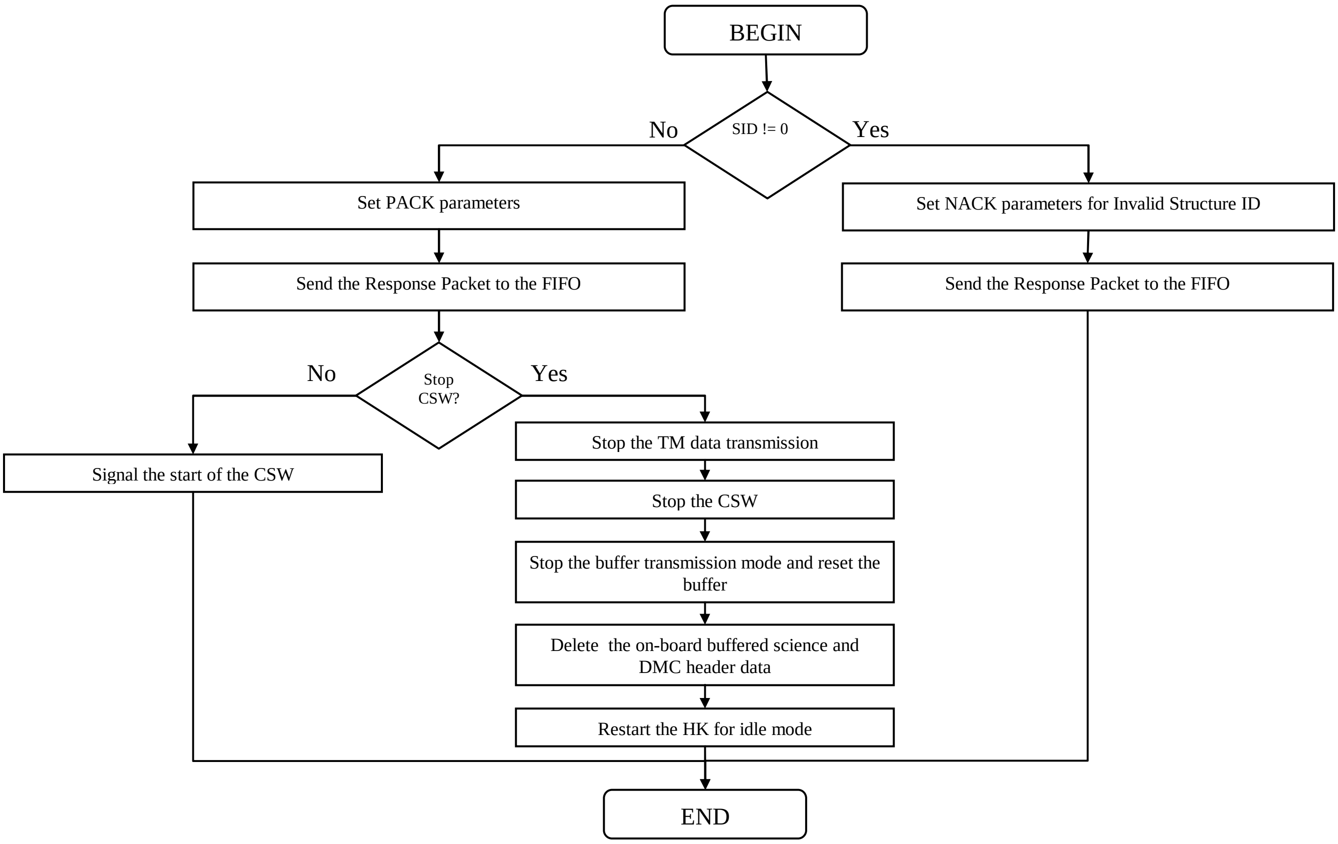


Figure 36. Flowchart of the Procedure for the Start and Stop Commands Handling

5.7.8.6 Data

Writes data to the FIFO.

5.7.8.7 Resources

None

5.7.9 ACT_TEST_PHOT/SPEC Command

5.7.9.1 Type

`Spu_tst.c` – procedure

5.7.9.2 Function

This procedure is used to start the SPU Test Mode for Spectroscopy or Photometry.

5.7.9.3 Interfaces

Call: KS_FIFOPut .... Sends the response packet to the FIFO (Data Transmission task)

Call: KS_SemaSignal(SPU_SPVS_TST)........ Signals the start of the Supervisor for the SPU Test

Call: KS_SemaSignal(SEMA_SPUTESTS) ... Signals the start of the data generator of photometry' frames

Call: KS_SemaSignal(SEMA_SPUTESTP) ... Signals the start of the data generator of spectroscopy' frames

5.7.9.4 Dependencies

Called from perform activity command module.

5.7.9.5 Processing

See flowchart in Figure 37.

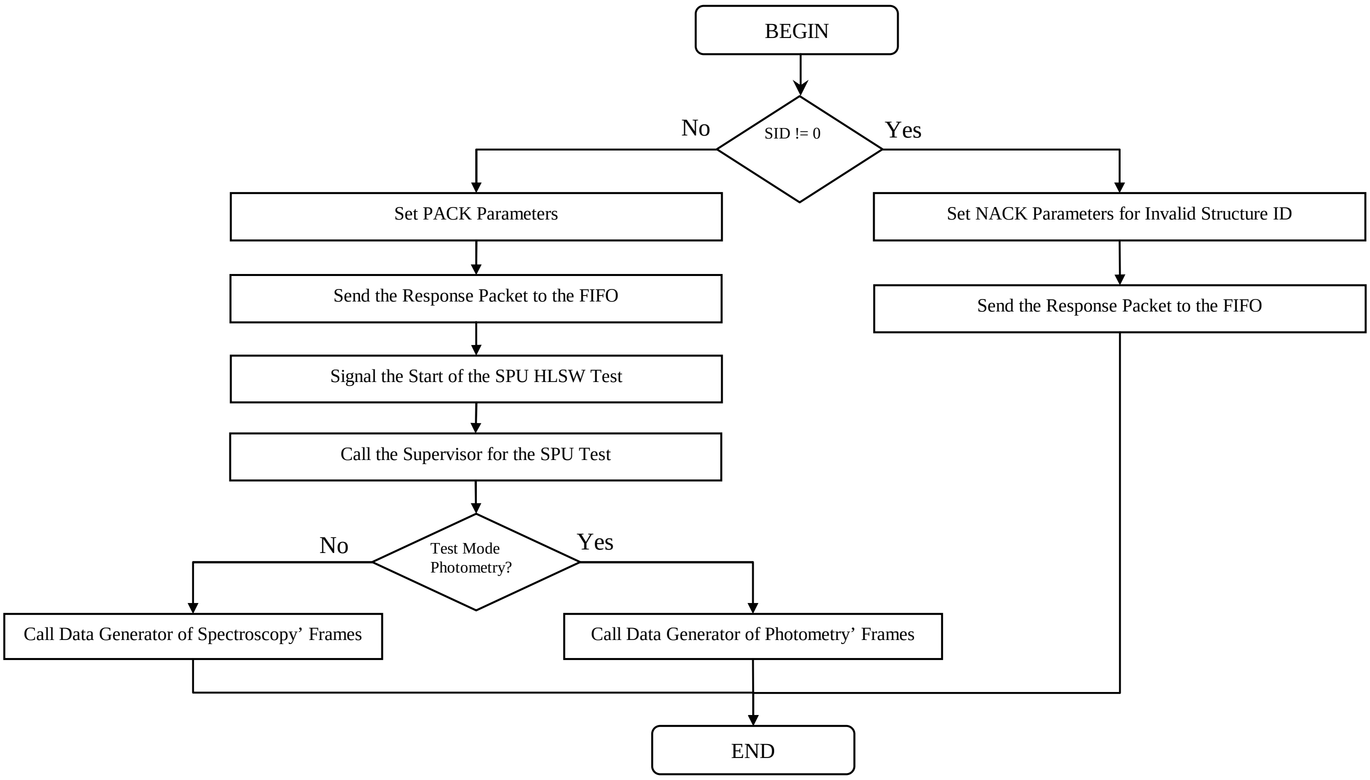


Figure 37. Flowchart of theProcedure for SPU Test Command Handling

5.7.9.6 Data

Writes data to the FIFO.

5.7.9.7 Resources

None

5.7.10 CONNECT_DMC Command

5.7.10.1 Type

`DMC_Con.c` – procedure

5.7.10.2 Function

This procedure acknowledges the reception of the Connect to DEC/MEC command and starts the link to DEC/MEC as Master or Slave depending on the attached parameter.

5.7.10.3 Interfaces

Call: PerformConnectToDmc.DEC/MEC Link Start Task

Call: KS_FIFOPut..................Sends the response packet to the FIFO (Data Transmission task)

Call: KS_TaskSleep ....Time delay

5.7.10.4 Dependencies

Called from perform activity command module.

5.7.10.5 Processing

See flowchart in Figure 38.

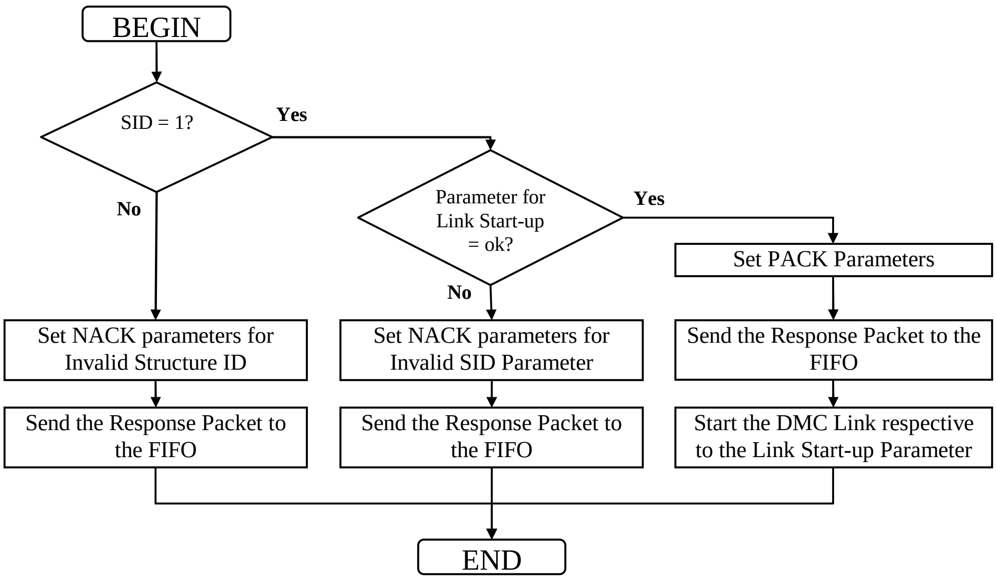


Figure 38. Flowchart of the Procedure for Connect to DEC/MEC Command Handling

5.7.10.6 Data

Writes data to the FIFO.

5.7.10.7 Resources

None

5.7.11 Write Command

5.7.11.1 Type

`write.c` – procedure

5.7.11.2 Function

This procedure is used to update the compression lookup tables (DXS, DET_CST_PHOT, DET_CST_SPEC and SIM_DATA tables).

It contains the following functions:

`InitDetectorSelectionTables` Initialises the Data Selection Table

`InitTables`.................................. Initialises the Bit Mask Tables

A short description of the ten-lookup tables is given below.

➢ **Write of Detectors Selection Table in the SPU Memory (WRT_DXS)**

Seven write commands allow the upgrade of the detectors tables from which data are requested. The detectors (photoconductors and bolometers) information is written in seven tables. Five for the bolometers and two for the photoconductors. The length of each DXS table is 96 Bytes (0x18 words).

The same HLSW is running in both SPUs Red and Blue. The SW accepts all seven write commands WRT_DXS1-7. The user has to choose the command to send depending on the Detector type (photometer WRT_DXS1-5 or spectrometer WRT_DXS6-7).

Both SPUs have the same set of commands, the SW accepts all commands but the usage of the tables depends on the observing mode (photometry or spectroscopy) and the wavelength (SWL and LWL).

- WRT_DXS1-5 commands are required for photometry data processing for both SPUs
- WRT_DXS6 command is actually required for spectroscopy for both SPUs
- WRT_DXS1-4 and WRT_DXS6 commands are required for SWL data processing in any SPU
- WRT_DXS5 and WRT_DXS6 commands are required for LWL data processing in any SPU
- WRT_DXS7 command may be used in degraded mode whenever one SPU sub-unit has to process data from both detectors (SWL and LWL). In the current SW version WRT_DXS7 is not exploited as the degraded mode is not implemented .

See RD016 (HLSW User Manual) for more details.

➢ **Write of Detectors Constants for Spectroscopy in the SPU Memory (WRT_DET_CST_SPEC)**

It allows the upgrade of the table of photoconductor constants, which are relevant for the pre-processing step in spectroscopy. The table length is 96 Bytes (0x18 words). See RD016 (HLSW User Manual) for more details.

- **Write of Detectors Constants for Photometry in the SPU Memory (WRT_DET_CST_PHOT)**

  It allows the upgrade of the table of bolometer constants, which are relevant for the pre-processing step in photometry. The table length is 96 Bytes (0x18 words). See RD016 (HLSW User Manual) for more details.

- **Write of Simulated Data Parameters in the SPU Memory (WRT_SIM_DATA)**

  It allows the upgrade of the parameters table, which will be used for data generation in the SPU test mode. The table length is 96 Bytes (0x18 words). See RD016 (HLSW User Manual) for more details.

5.7.11.3 Interfaces

Call: InitDetectorSelectionTables ..... Initialises the Data Selection Table

Call: KS_FIFOPut .... Sends the response packet to the FIFO (Data Transmission task)

5.7.11.4 Dependencies

Called from watch process.

5.7.11.5 Processing

See flowchart in Figure 39.

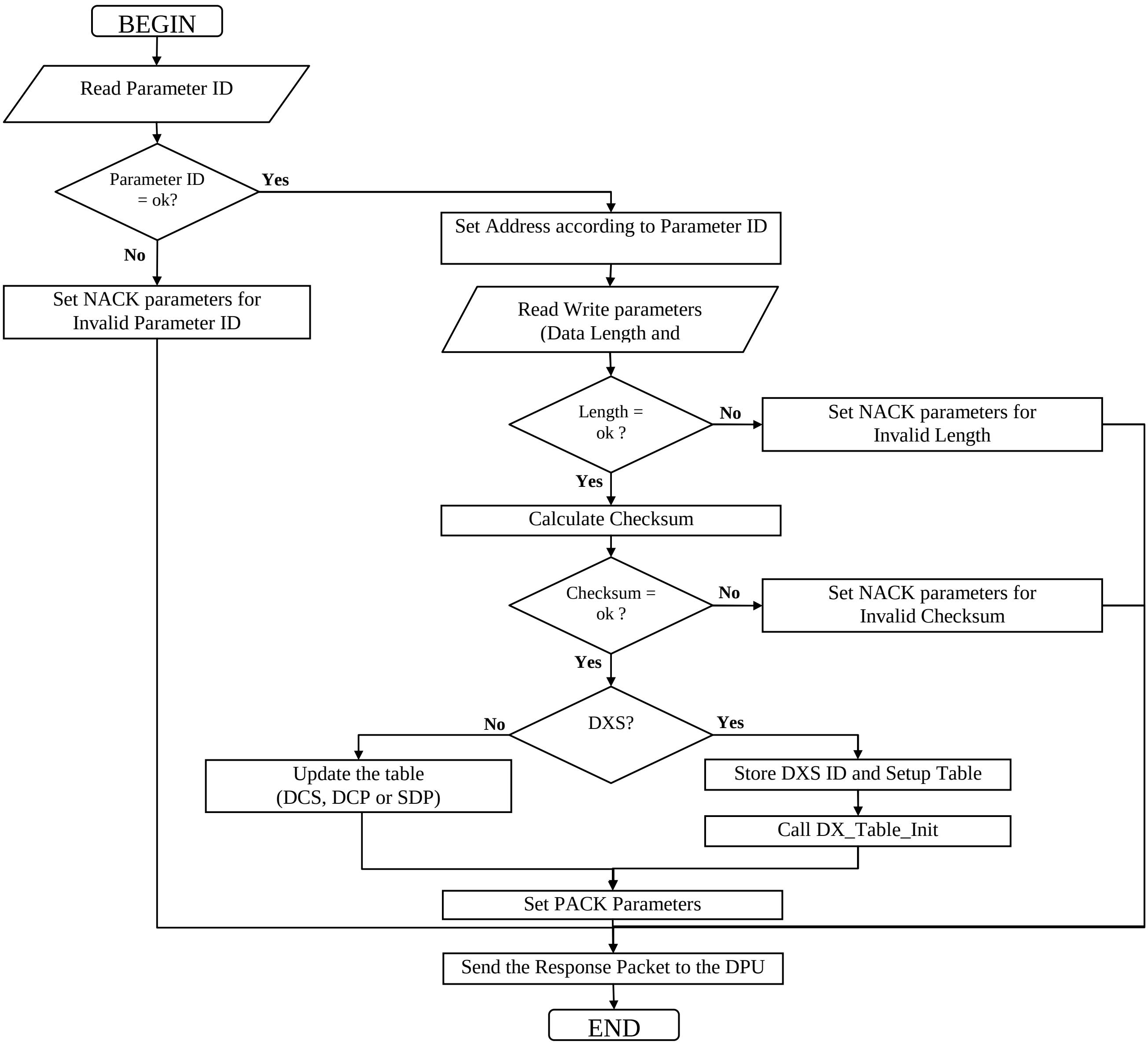


Figure 39. Flowchart of the Procedure for Write Command Handling

5.7.11.6 Data

Writes data to the FIFO and the data memory.

5.7.11.7 Resources

None

5.8 Compression Software

5.8.1 Compression Software Main Task

5.8.1.1 Type

`csw.c` – task

5.8.1.2 Function

The compression software compresses the science data according to the required compressed mode. The compressed parameter is received via the supervisor. The compression software will accordingly start appropriate tasks.

5.8.1.3 Interfaces

Call: PhotoReductionAndCompression..... Bol1 and Bol2 compression procedure

Call: PhotoLosslessCompressionMode ..... Bol3 compression procedure

Call: PhotoTransparentMode .... Bol4 compression procedure

Call: PhotoBufferTransmissionMode........ Procedure for Buffer Transmission Mode in Photometry

Call: SpectroReductionAndCompressionMode..Spec1 and Spec2 compression procedure

Call: SpectroLosslessCompressionMode .. Spec3 compression procedure

Call: SpectroTransparentMode.................. Spec4 compression procedure

Call: SpectroBufferTransmissionMode..... Procedure for Buffer Transmission Mode in Spectroscopy

Call: KS_TaskSleep .... Time delay

Call: KS_SemaSignal .... Signals the start of HK transmission

5.8.1.4 Dependencies

Called from the main task.

Activated by Supervisor or Supervisor_Test task.

5.8.1.5 Processing

See flowchart in Figure 40 and Figure 41.

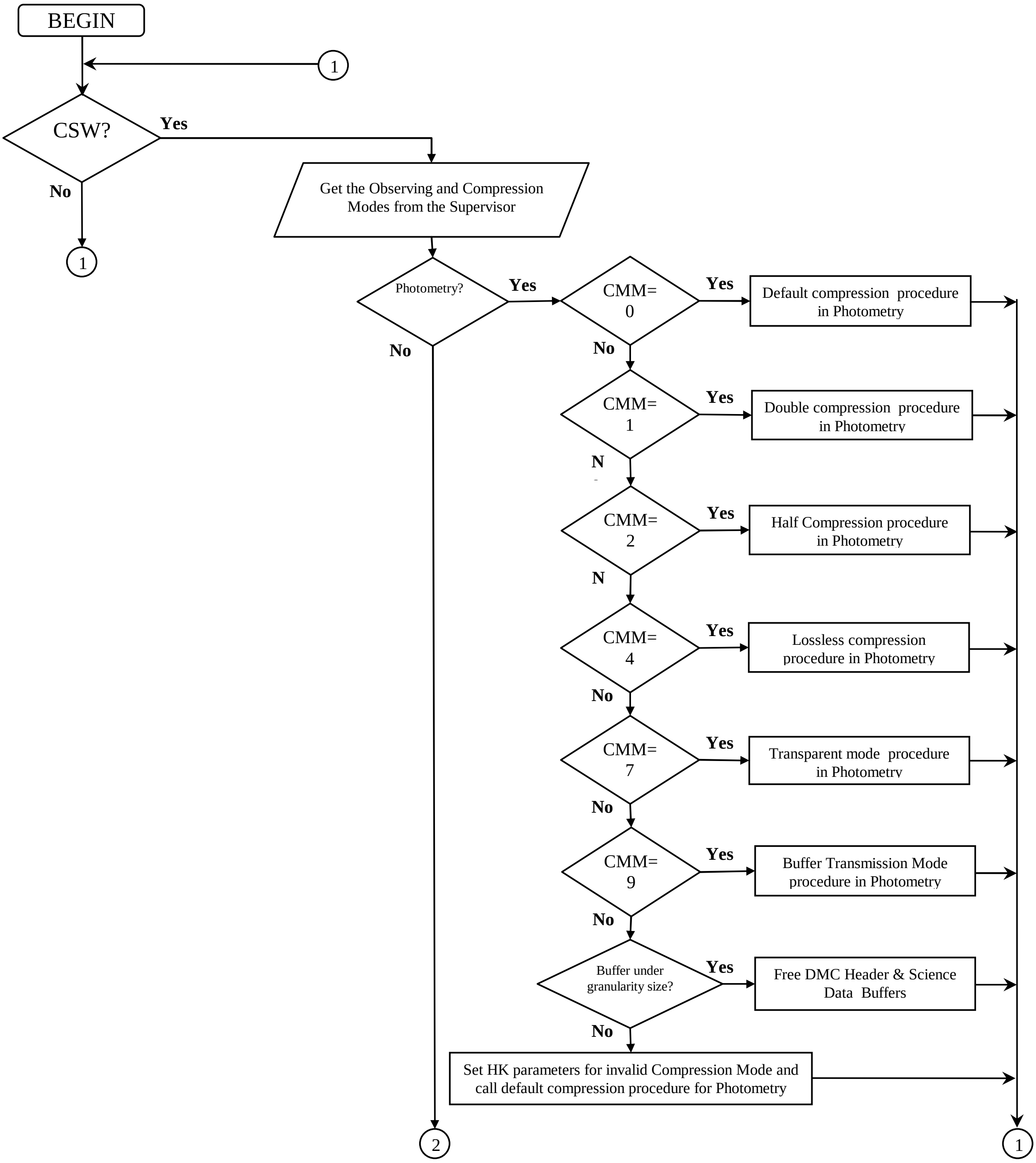


Figure 40. Flowchart of the Compression Software Main Task (Part 1)

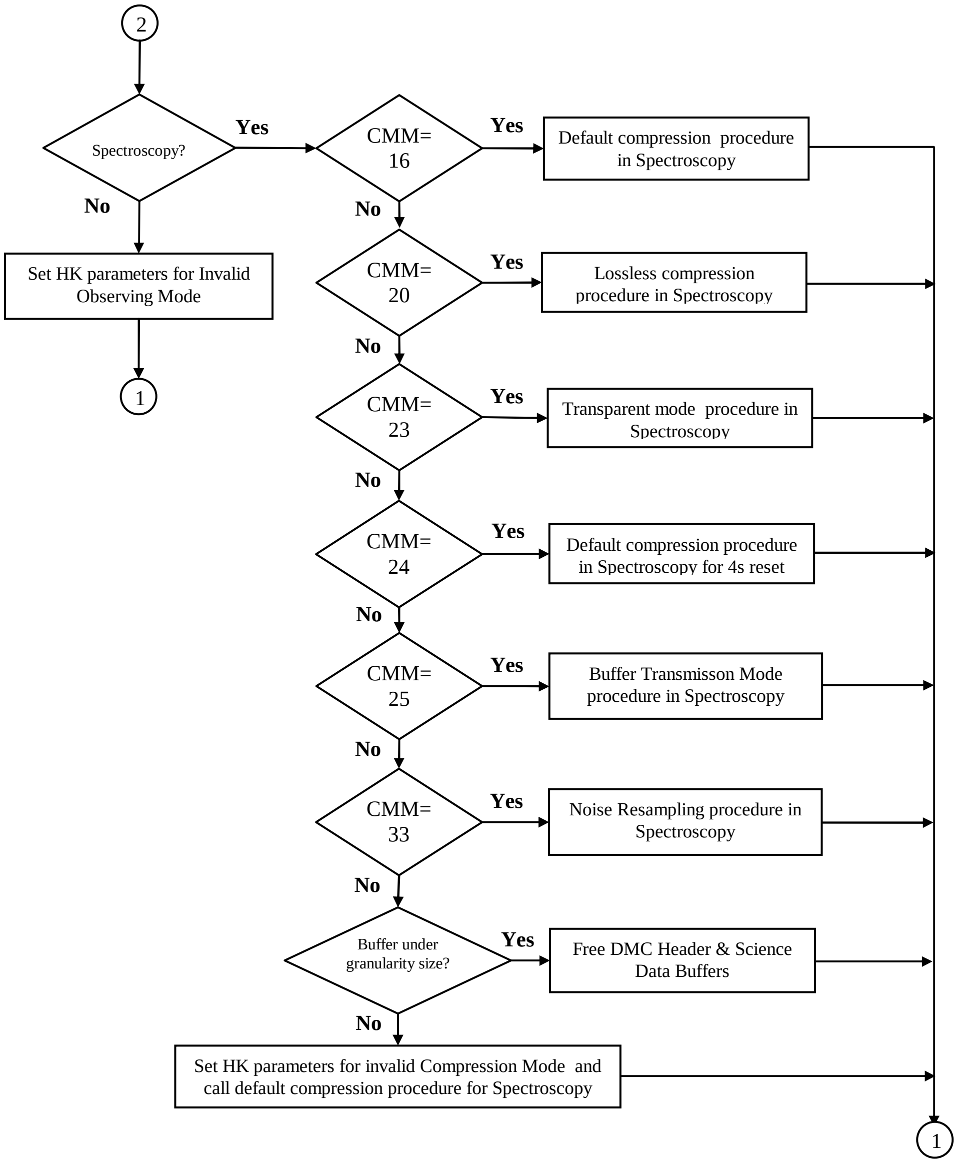


Figure 41. Flowchart of the Compression Software Main Task (Part 2)

5.8.1.6 Data

None

5.8.1.7 Resources

None

5.8.2 Photometry Modes 0, 1 and 2

5.8.2.1 Type

`bol_ex.c` – procedure

5.8.2.2 Function

This procedure performs data compression in photometry for the Bol1 or Bol2 modes depending on the received compression parameter.

* Bol1 (Default Compression Mode): This is the default mode for photometry.
* Bol2 (Double the Compression Ratio): This mode is used whenever the default compression ratio should be doubled.
* Bol0 (Half Compression Mode): This mode is used whenever the readout rate in photometry is decreased to 20 Hz. It performs 2 sample averaging for reduction.

5.8.2.3 Interfaces

Call: DmcHeaderCompression ..DMC header extraction and compression

Call: Dxs....................................Detector selection

Call: PreProcessing ....Pre-processing of the data

Call: FloatAveraging ....when averaging arbitrary number of samples

Call: Averaging4Samples ....default averaging

Call: PhotoCompression............Temporal and spatial redundancy reduction, lossless coding

Call: PhotoCompression2..........Temporal and spatial redundancy reduction, lossless coding

Call: BuildCompressedEntity ....Write the compressed data into the output buffer

5.8.2.4 Dependencies

Called from the compression software main task.

5.8.2.5 Processing

See flowchart in *Figure 42*.

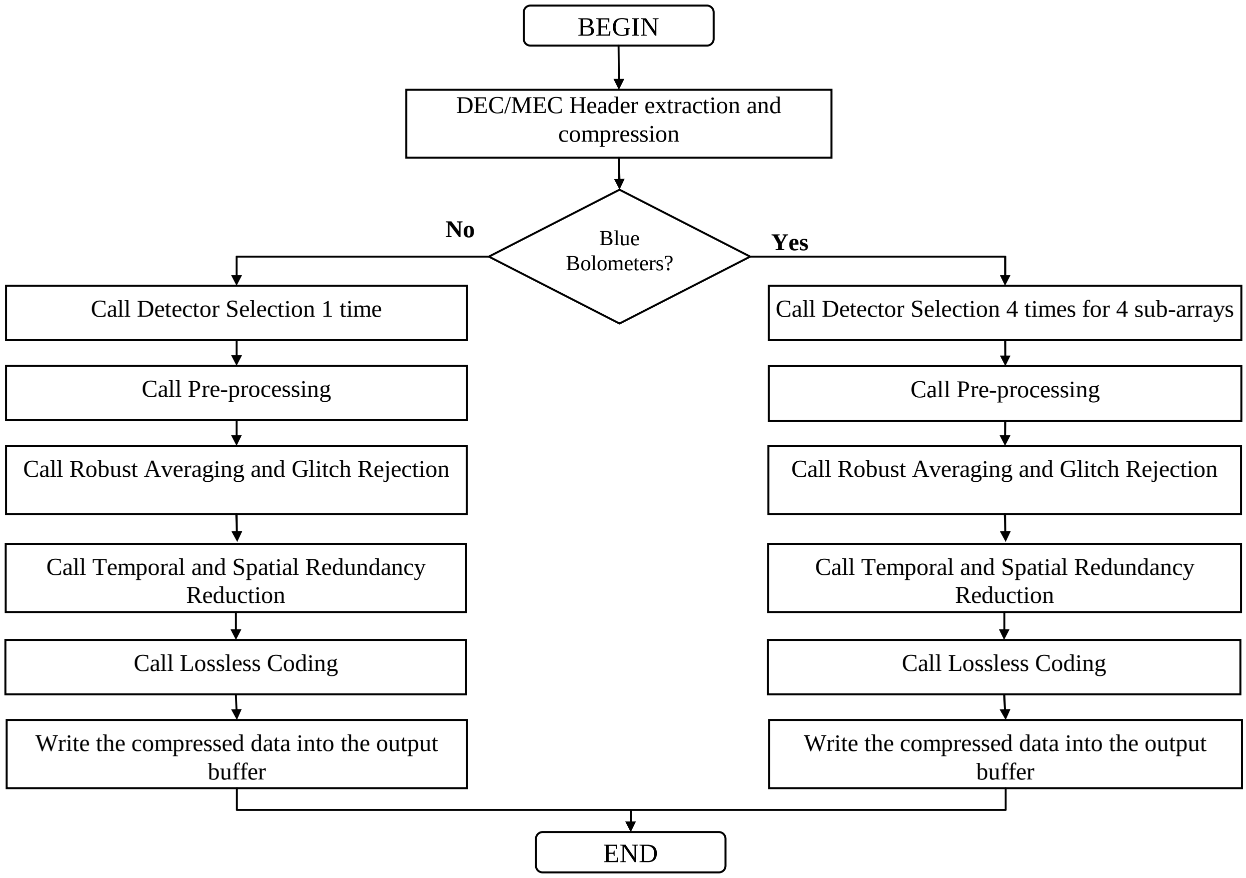


Figure 42. Flowchart of the Bol_ex Procedure

5.8.2.6 Data

None

5.8.2.7 Resources

None

5.8.3 Photometry Mode 3

5.8.3.1 Type

`bol3.c` – procedure

5.8.3.2 Function

Bol3 (Lossless Compression Mode) is used for the lossless compression of photometry data from selected detectors.

5.8.3.3 Interfaces

Call: DmcHeaderCompression.............DMC header extraction and compression

Call: Dxs...Detector selection

Call: TSReductionForPhotoLlc_..........Temporal and spatial redundancy reduction

Call: LosslessCompressionForPhoto....Lossless Coding

Call: BuildCompressedEntity..............Write the compressed data into the output buffer

5.8.3.4 Dependencies

Called from the compression software main task.

5.8.3.5 Processing

See flowchart in *Figure 43*.

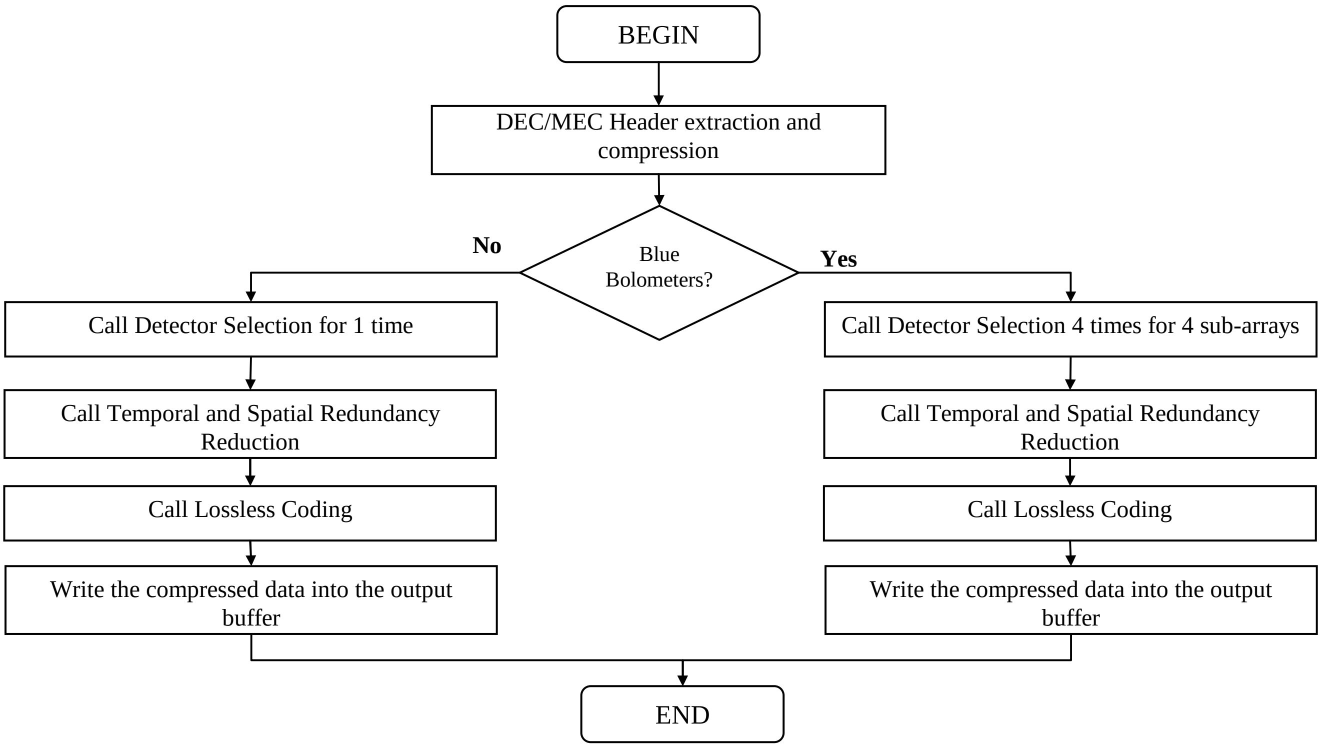


Figure 43. Flowchart of the Bol3 Procedure

5.8.3.6 Data

None

5.8.3.7 Resources

None

5.8.4 Photometry Mode 4

5.8.4.1 Type

`bol4.c` – procedure

5.8.4.2 Function

Bol4 (Transparent Mode) is used for the transparent mode. It transmits raw data from selected detectors without compression.

5.8.4.3 Interfaces

Call: DmcHeaderCompression..DMC header extraction and compression

Call: Dxs...................................Detector selection

Call: BuildCompressedEntity....Write the compressed data into the output buffer

5.8.4.4 Dependencies

Called from the compression software main task.

5.8.4.5 Processing

See flowchart in *Figure 44*.

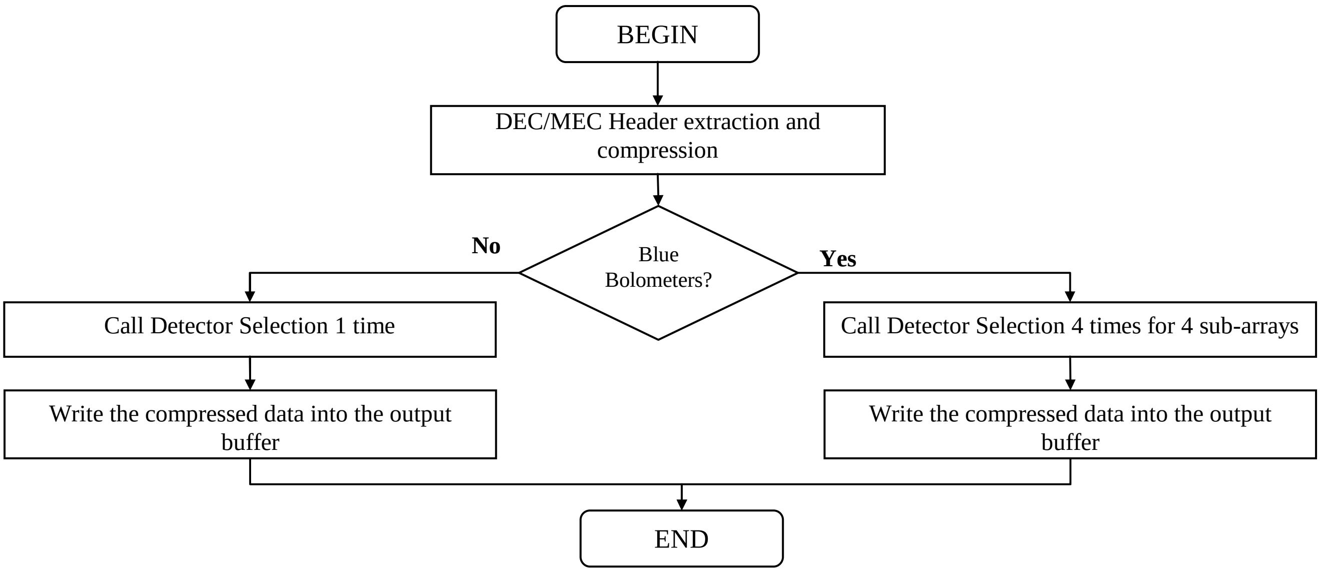


Figure 44. Flowchart of the Bol4 Procedure

5.8.4.6 Data

None

5.8.4.7 Resources

None

5.8.5 Photometry Buffer Transmission Mode

5.8.5.1 Type

`buf_bol.c` – procedure

5.8.5.2 Function

This procedure deals with the Buffer Transmission mode in photometry. It fills the memory for a certain time with the data, then sends them in raw mode.

BUF_Bol (Buffer Transmission Mode): The entire memory buffers will be filled with DEC/MEC header and science data (Blue SPU: 5 x 3 seconds; Red SPU: 5 x 12 seconds). After 15/60 seconds (Blue/Red SPU) the SPU HLSW ignores the input DEC/MEC data and start sending the buffered data for duration of 150 seconds in case of Blue SPU and 155 seconds in case of Red SPU. The data is then transmitted to the DPU with rate compatible to the telemetry rate in burst mode. The telemetry packets are transmitted at 60 msec average time intervals to DPU i.e. 37 TM packets are made ready for DPU every 2220 msec

In **SWL** the buffers will be filled with 15 seconds data from all detectors. This data will be stored in five buffers (buffer1, buffer2, buffer3, buffer4 and buffer5). Therefore, each buffer includes 3 seconds of data. Then the SPU HLSW ignores all incoming data from DMC and begins to transmit the buffered 15 seconds data to DPU. They will be sent in the following sequence: buffer5, buffer1, buffer2, buffer3 and then buffer4. The data consists of uncompressed DMCH (only from 1 array ID) and uncompressed science data. In the compressed entity header the CDHS is set to Zero while SCIS contains the full data size (size = 0x1E618 words = 497760 Bytes). During this period HK are transmitted at regular rate to DPU.

- Blue SPU: 5 x 498 packets = 2490 packets of 1000 Bytes are sent at average time interval of 50 ms
- SPU puts every 2220 msec 37 TM packets to DPU (2220/37=60 ms time interval between two TM packets).
- 28 B + 497760 B (SD & DMH) + 24 B (TM Header) x 498 packets = 509740 Bytes x5 = 2548700Bytes
- 2548700Bytes*60ms/1024Bytes = 149,4sec (150sec)
- 2548700Bytes /(150+15) sec = 121 kbit/s
- The averaged TM rate for the blue SPU is 121 kbit/s

In **LWL** the buffers will be filled with 60 seconds data from all detectors. This data will be stored in five buffers (buffer1, buffer2, buffer3, buffer4 and buffer5). Therefore, each buffer includes 12 seconds of data. Then the SPU HLSW ignores all incoming data from DMC and begins to transmit the buffered 60 seconds to DPU. They will be sent in the following sequence: buffer5, buffer1, buffer2, buffer3 and then buffer4. The data consists of uncompressed DMCH and uncompressed science data. In the compressed entity header the CDHS is set to Zero while SCIS contains the full data size (size = 0x1F860 word = 516480 Bytes). During this period HK are transmitted at regular rate to DPU.

- Red SPU: 5 x 517 packets = 2585 packets of 1000 Bytes are sent at average time interval of 50 ms
- SPU puts every 2220 msec 37 TM packets to DPU (2220/37=60 ms time interval between two TM packets).
- 28 B + 516480 B (SD & DMH) + 24 B x 517 packets = 528916 Bytes x5 = 2644580 Bytes
- 2644580 Bytes*60ms/1024Bytes = 154.9 sec (155sec)
- 2644580 Bytes/(155+60) sec = 90kbit/s
- The averaged TM rate for the red SPU is 90 kbit/s

- The averaged TM rate for both SPUs is 211 kbit/s.

5.8.5.3 Interfaces

Call: ReadInCircularBuffer .... Reads the science data from input buffer

Call: BuildCompressedEntity .... Write the compressed data into the output buffer

5.8.5.4 Dependencies

Called from the compression software main task.

5.8.5.5 Processing

See flowchart in Figure 45.

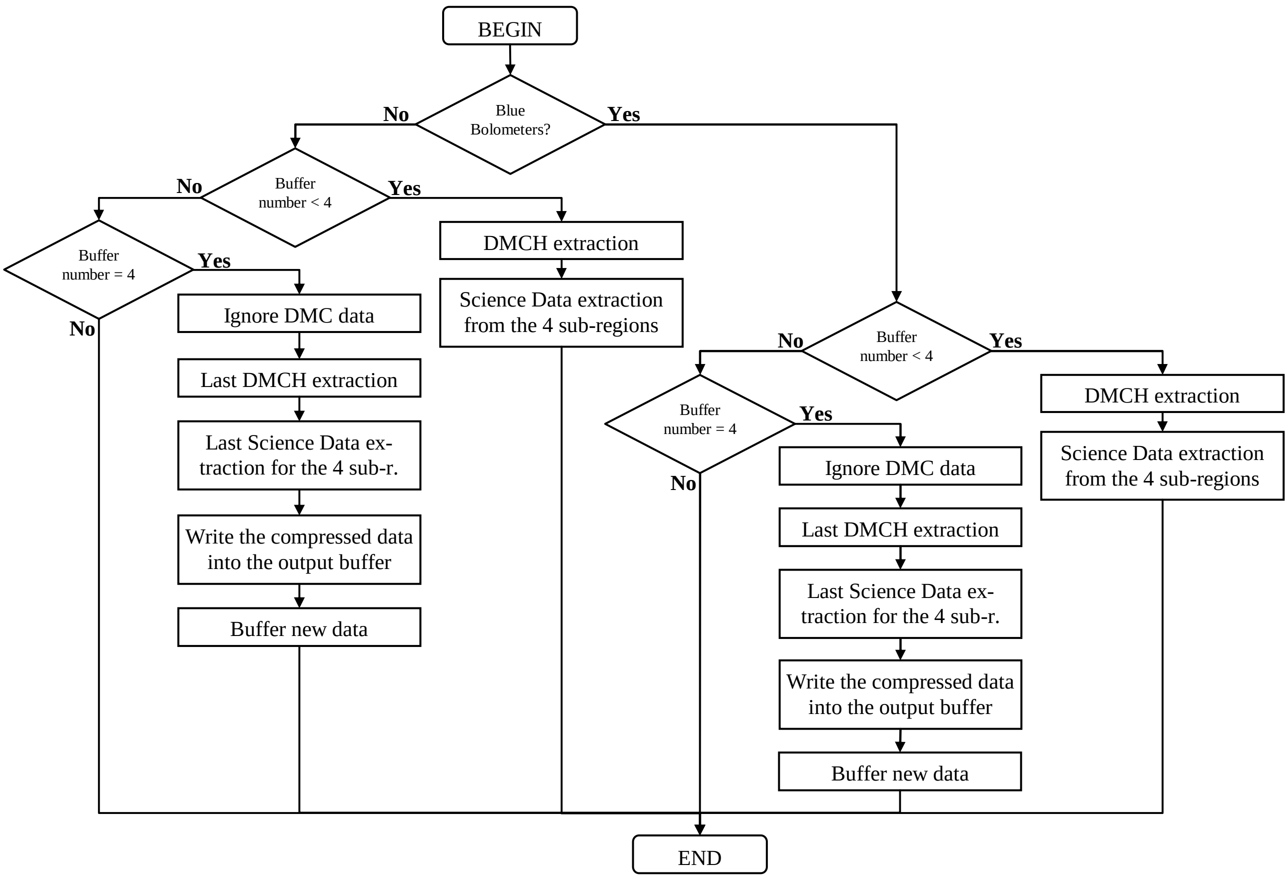


Figure 45. Flowchart of the BUF_Bol Procedure

5.8.5.6 Data

None

5.8.5.7 Resources

It requires at least 3 MB of memory.

5.8.6 Spectroscopy Modes 1 and 2

5.8.6.1 Type

`spec1_2.c` – procedure

5.8.6.2 Function

This procedure performs data compression in spectroscopy for the Spec1, Spec2 or Spec5 modes depending on the received compression parameter.

* Spec1 (Default Compression Mode): This is the default mode for spectroscopy.
* Spec2 (Double the Compression Ratio): This mode is used whenever the default compression ratio should be doubled.

The Spec5 mode performs a default compression mode for a 4-second ramp. The ramp is subdivided to 2 sub-ramps of 2-second length.

* Spec5 (Compression Mode for 4s Reset): This mode is used to handle detectors data with a reset interval of 4s. The compression software receives 1024 frames and put them into two buffers. Each buffer contains 512 frames. Then the SW resamples the data from 256Hz to 8Hz with the implemented fitting algorithm.

5.8.6.3 Interfaces

Call: DmcHeaderCompression..DMC header extraction and compression

Call: Dxs...................................Detector selection

Call: PreProcessing ....Pre-processing of the data

Call: RampFit ....Ramp fitting and glitch rejection

Call: TSReductionForSpectro ...Resorting the data and Temporal and spatial redundancy reduction

Call: LosslessCompressionForSpectro ....Lossless Coding

Call: BuildCompressedEntity ....Write the compressed data into the output buffer

5.8.6.4 Dependencies

Called from the compression software main task.

5.8.6.5 Processing

See flowchart in Figure 46.

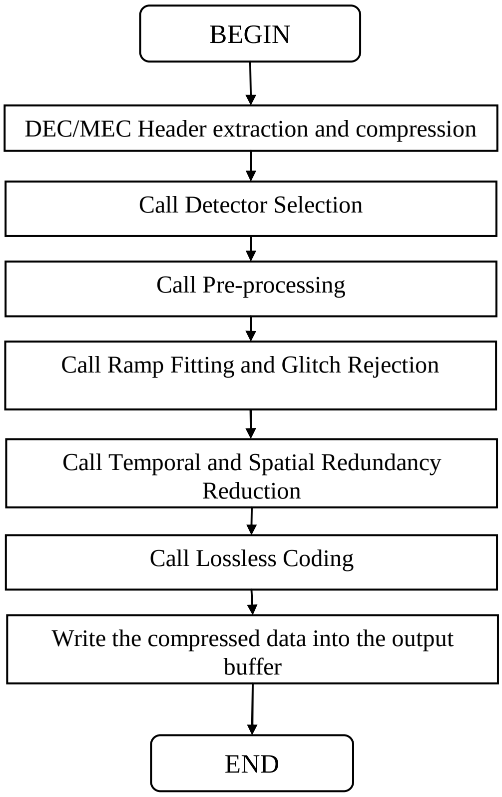


Figure 46. Flowchart of the Spec1_2 Procedure

5.8.6.6 Data

None

5.8.6.7 Resources

None

5.8.7 Spectroscopy Mode 3

5.8.7.1 Type

`spec3.c` – procedure

5.8.7.2 Function

Spec3 (Lossless Compression Mode) is used for the lossless compression of spectroscopy data from selected detectors.

5.8.7.3 Interfaces

Call: DmcHeaderCompression..DMC header extraction and compression

Call: Dxs...................................Detector selection

Call: TSReductionForSpectro ...Resorting the data and Temporal and spatial redundancy reduction

Call: LosslessCompressionForSpectro Lossless Coding

Call: BuildCompressedEntity....Write the compressed data into the output buffer

5.8.7.4 Dependencies

Called from the compression software main task.

5.8.7.5 Processing

See flowchart in Figure 47.

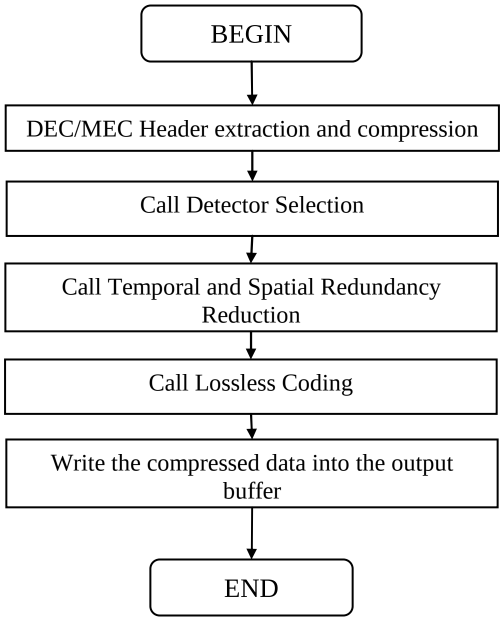


Figure 47. Flowchart of the Spec3 Procedure

5.8.7.6 Data

None

5.8.7.7 Resources

None

5.8.8 Spectroscopy Mode 4

5.8.8.1 Type

`spec4.c` – procedure

5.8.8.2 Function

Spec4 (Transparent Mode) is used for the transparent mode. It transmits raw data from selected detectors without compression.

5.8.8.3 Interfaces

Call: DmcHeaderCompression..DMC header extraction and compression

Call: Dxs....................................Detector selection

Call: BuildCompressedEntity....Write the compressed data into the output buffer

5.8.8.4 Dependencies

Called from the compression software main task.

5.8.8.5 Processing

See flowchart in Figure 48.

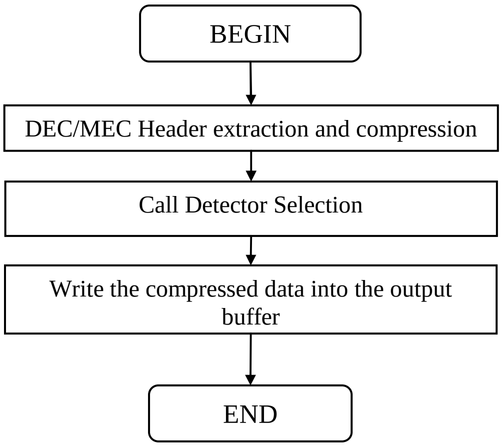


Figure 48. Flowchart of the Spec4 Procedure

5.8.8.6 Data

None

5.8.8.7 Resources

None

5.8.9 Spectroscopy Buffer Transmission Mode

5.8.9.1 Type

`buf_spec.c` – procedure

5.8.9.2 Function

This procedure deals with the Buffer Transmission mode in spectroscopy. It fills the memory for a certain period with data, and then sends them in raw mode.

BUF_Spec (Buffer Transmission Mode): The entire memory buffers will be filled with DEC/MEC header and science data (Blue and Red SPU: 5 x 2 seconds). After 10 seconds the SPU HLSW ignores the input DEC/MEC data and start sending the buffered data for duration of 152 seconds. The data is then transmitted to the DPU with rate compatible to the telemetry rate in burst mode. The telemetry packets are transmitted at 60 msec average time intervals to DPU i.e. 37 TM packets are made ready for DPU every 2220 msec.

In **LWL** and **SWL** the buffers will be filled with 10 seconds data from all detectors. This data will be stored in five buffers (buffer1, buffer2, buffer3, buffer4 and buffer5). Therefore, each buffer includes 2 seconds of data. Then the SPU HLSW ignores all incoming data from DMC and begins to transmit the buffered 10 seconds data to DPU. They will be sent in the following sequence: buffer5, buffer1, buffer2, buffer3 and then buffer4. The data consists of uncompressed DMCH and uncompressed science data. In the compressed entity header the CDHS is set to Zero while SCIS contains the full data size (size = 0x1EE00 words = 494 kB). During this period HK are transmitted at regular rate to DPU.

- Blue/Red SPU: 5 buffer x 506 packets = 2530 packets of 1000 Bytes are sent at average time interval of 60 ms.
- SPU puts every 2220 msec 37 TM packets to DPU (2220/37=60 ms time interval between two TM packets).
- 28 B + 494 KB (SD & DMH) + 24 B (TM packet header) x 506 packets = 518028 Bytes x5 = 2590140 Bytes
- 2590140 Bytes /1024B*60ms = 151.77sec (152sec)
- 2590140/(152+10) sec = 124.91kbit/s

The averaged TM rate per SPU board is 124.91 kbit/s.

The averaged TM rate for both SPUs is 249,82 kbit/s.

5.8.9.3 Interfaces

Call: ReadInCircularBuffer ....Reads the science data from input buffer

Call: BuildCompressedEntity ....Write the compressed data into the output buffer

5.8.9.4 Dependencies

Called from the compression software main task.

5.8.9.5 Processing

See flowchart in Figure 49.

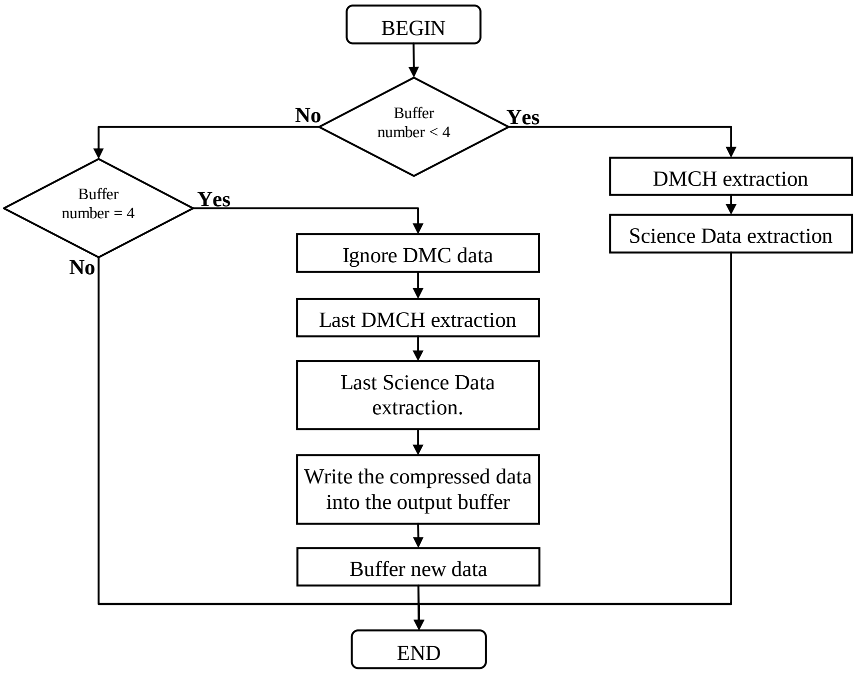


Figure 49. Flowchart of the BUF_Spec Procedure

5.8.9.6 Data

None

5.8.9.7 Resources

It requires at least 3 MB of memory.

5.9 Compression SW Modules

5.9.1 Detector Selection

5.9.1.1 Type

`dxs.c` – procedure

5.9.1.2 Function

This procedure performs the data selection according to the detector tables. The detectors selection tables are directly set by DPU in the SPU memory using a write command. This is used to identify the selected detectors from which data are solicited.

The solicited data are transmitted to DPU using the predefined compression mode. If the compression mode requires data reduction (compression modes spec1_2, spec5 or bol_ex), additional raw data from few channels can be transmitted lossless-compressed to fill the TM bandwidth.

5.9.1.3 Interfaces

None

5.9.1.4 Dependencies

Called from Bol_ex, Bol3, Bol4, Spec1_2, Spec3 or Spec4 procedure.

5.9.1.5 Processing

Detector selection procedure is depicted in Figure 50.

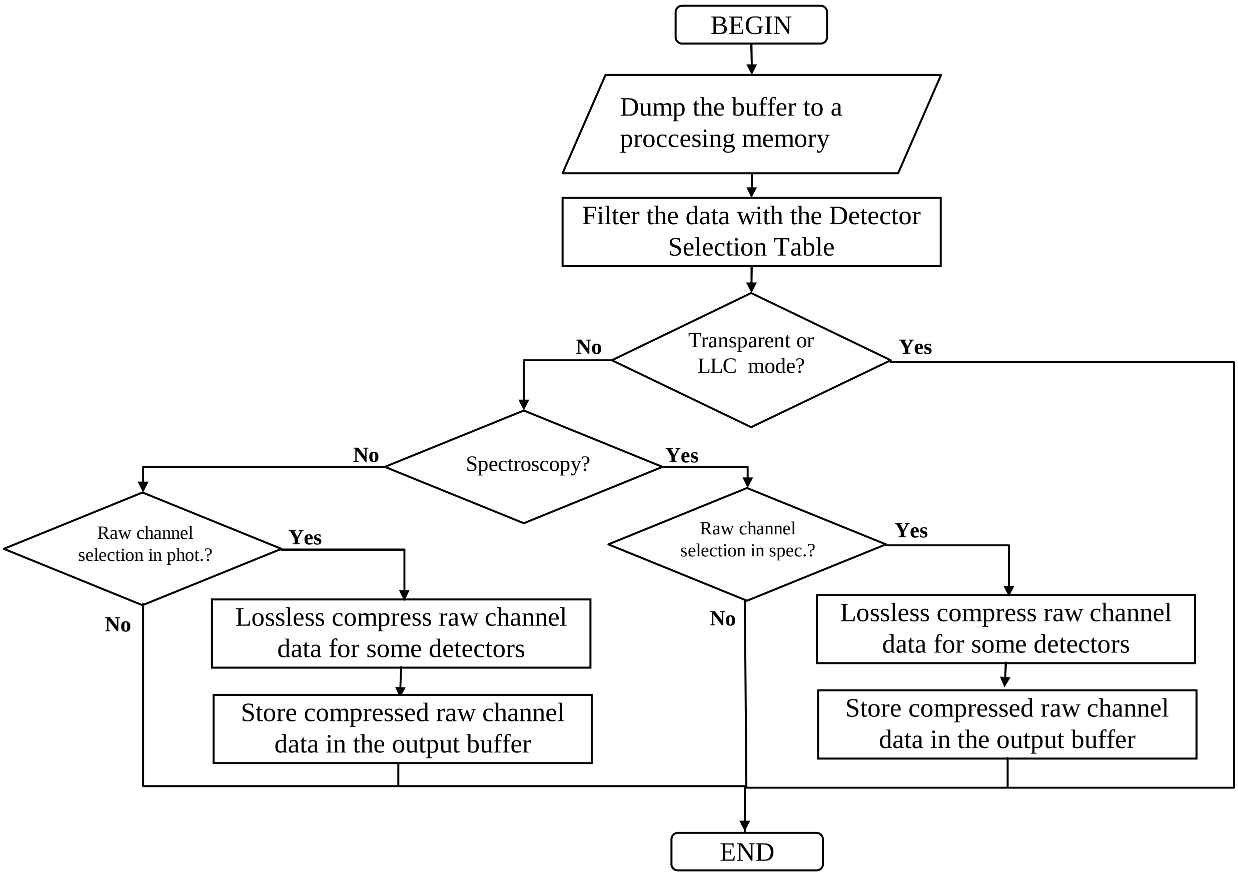


Figure 50. Flowchart of the Detector Selection Procedure

The currently implemented raw channel data compression algorithm is described in Appendix A.1.3.

5.9.1.6 Data

None

5.9.1.7 Resources

2 MB data memory

5.9.2 Preprocessing

5.9.2.1 Type

`p_proc.c` – procedure

5.9.2.2 Function

This procedure is used in photometry and spectroscopy mode in order to transform the received signal to the appropriate form (linear ramps in spectroscopy and constant signal in photometry). In fact, the use of this module is still TBD depending on the efficiency of the electronics and the electric filters.

5.9.2.3 Interfaces

None

5.9.2.4 Dependencies

Called from Bol_ex or Spec1_2 procedure.

5.9.2.5 Processing

See flowchart in Figure 51.

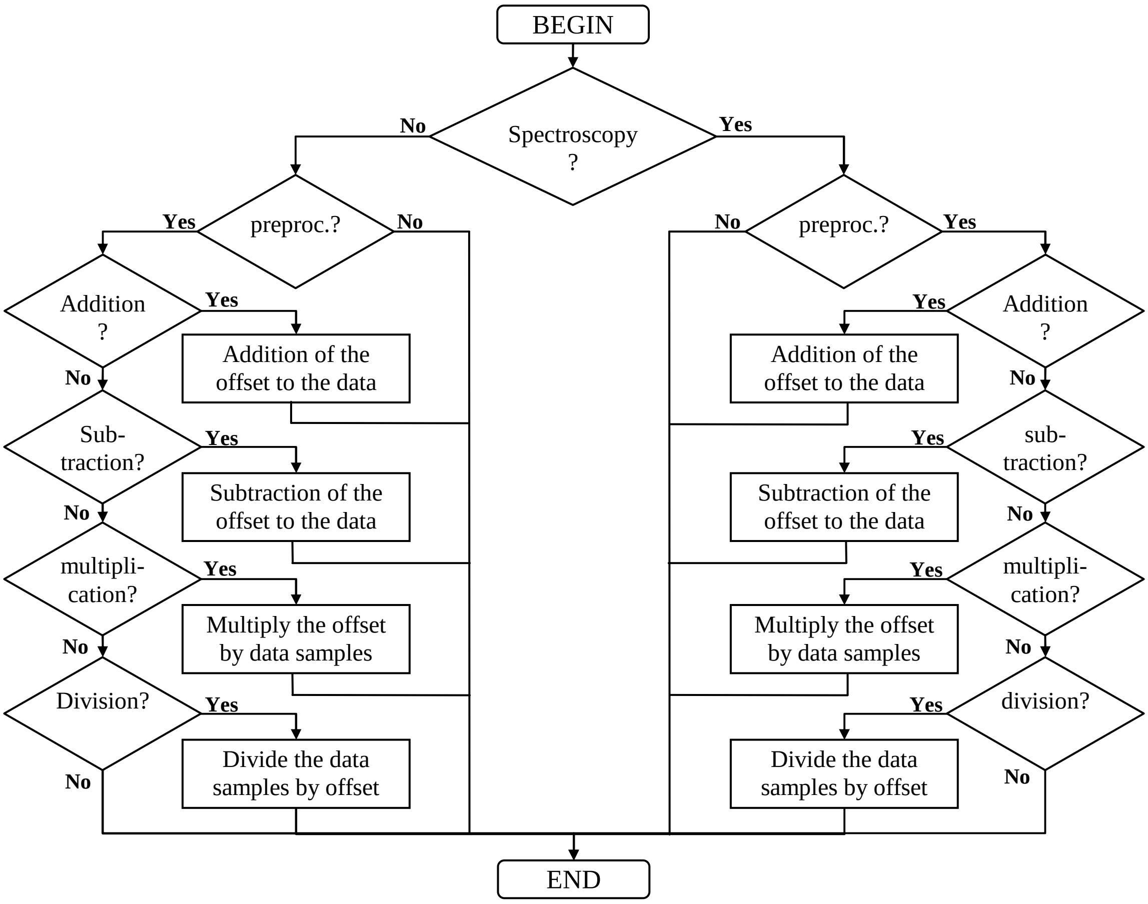


Figure 51. Flowchart of the Preprocessing Procedure

5.9.2.6 Data

None

5.9.2.7 Resources

None

5.9.3 Robust Averaging

5.9.3.1 Type

`average.c` – procedure

5.9.3.2 Function

This procedure is used to perform glitch rejection and oversampling reduction of the photometry data.

5.9.3.3 Interfaces

None

5.9.3.4 Dependencies

Called from Bol_ex procedure.

5.9.3.5 Processing

See flowchart in Figure 52.

A library of fitting algorithms for photometry data is proposed in Appendix B.2.1.

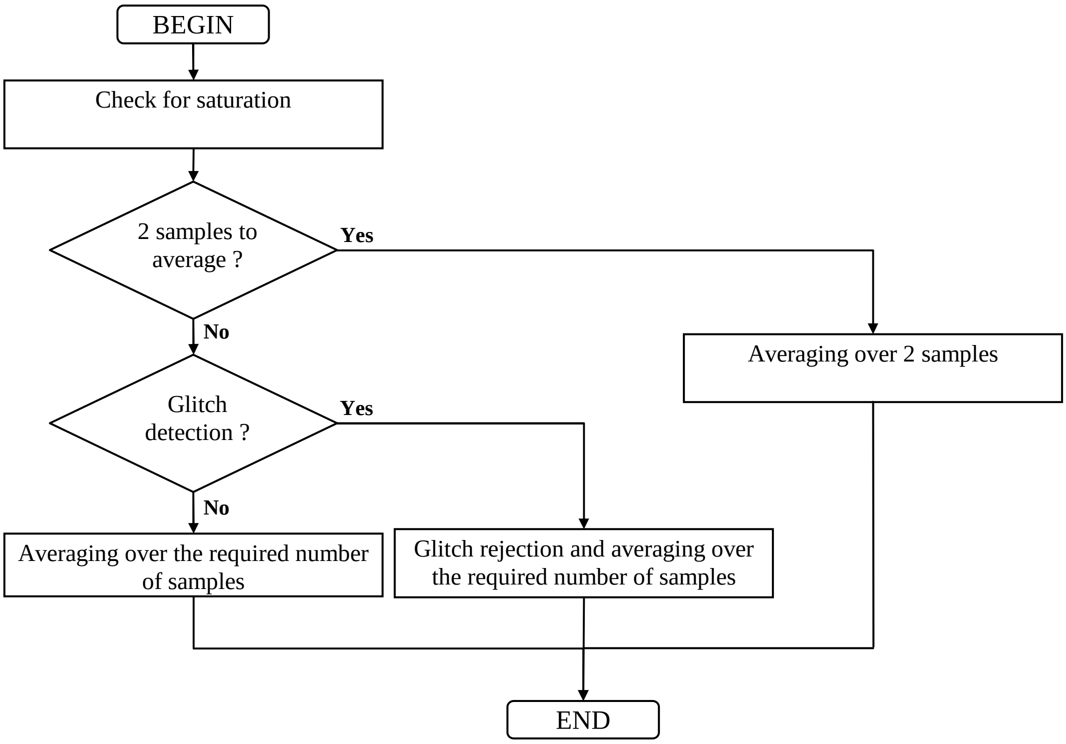


Figure 52. Flowchart of the Robust Averaging Procedure

The currently implemented averaging algorithm is described in Appendix B.1.1.

5.9.3.6 Data

None

5.9.3.7 Resources

None

5.9.4 Ramp Fitting

5.9.4.1 Type

`ramp_ft.c` – procedure

5.9.4.2 Function

This procedure is used to perform glitch rejection and calculating a slope or a mean out of a predefined set of samples in spectroscopy. The number of samples to use can be set using the write detectors constants in spectroscopy command (Section 5.7.11). From one to several slopes/means can be derived out of a ramp depending on the set parameter (Number of sample per sub-ramp)

5.9.4.3 Interfaces

None

5.9.4.4 Dependencies

Called from Spec1_2 procedure.

5.9.4.5 Processing

See the following flowchart.

A library of fitting algorithms for spectroscopy data is proposed in Appendix B.2.1.

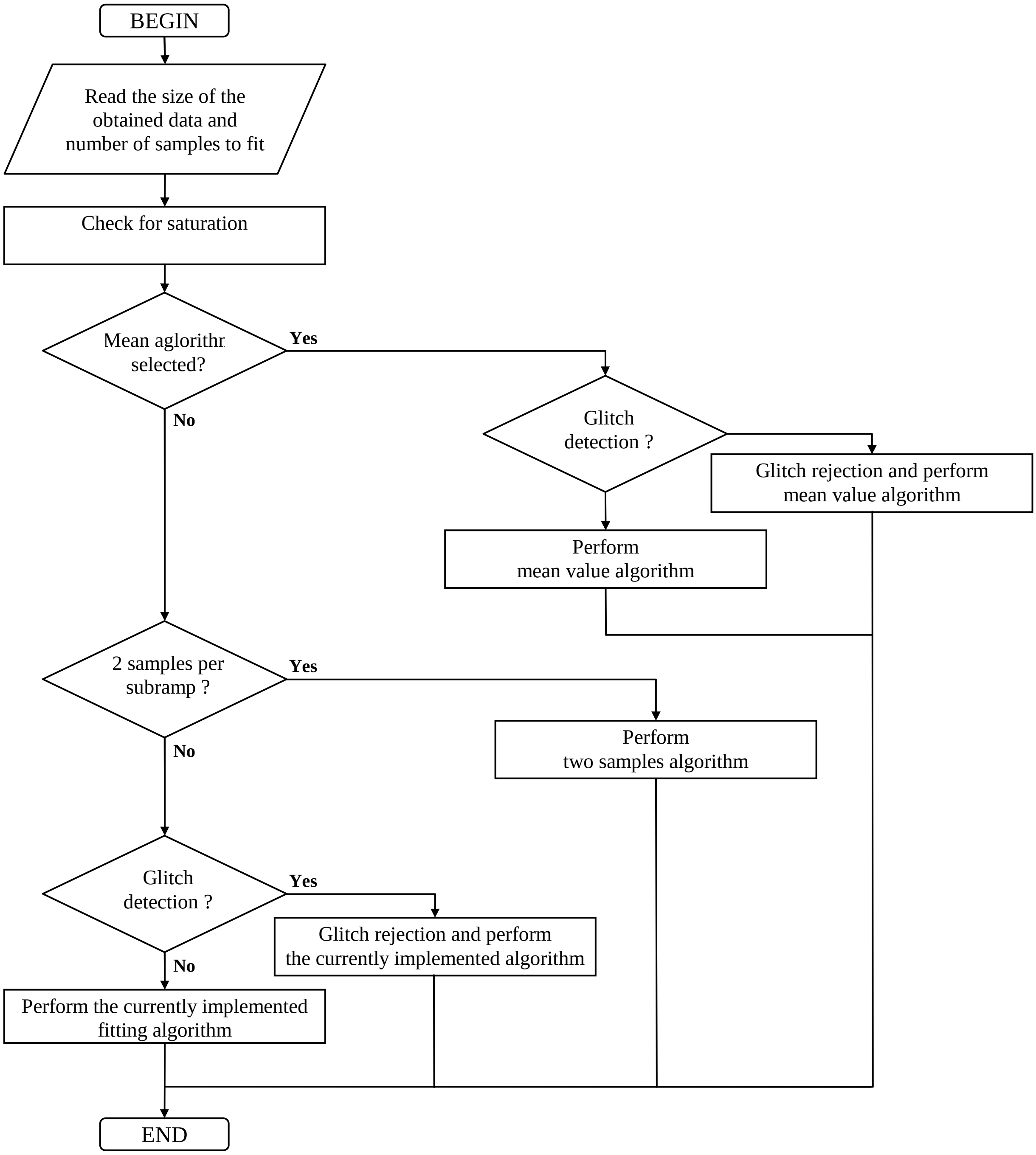


Figure 53. Flowchart of the Ramp-fitting Procedure

The currently implemented ramp-fitting algorithm is described in Appendix B.1.2.

5.9.4.6 Data

None

5.9.4.7 Resources

None

5.9.5 **Integration** (obsolete)

5.9.5.1 Type

`integ.c` – procedure

5.9.5.2 Function

This procedure performs on-board integration over the sensor readings in order to achieve the desired compression ratio.

5.9.5.3 Interfaces

None

5.9.5.4 Dependencies

Called from Bol_ex or Spec1_2 procedure.

5.9.5.5 Processing

See flowchart in Figure 54.

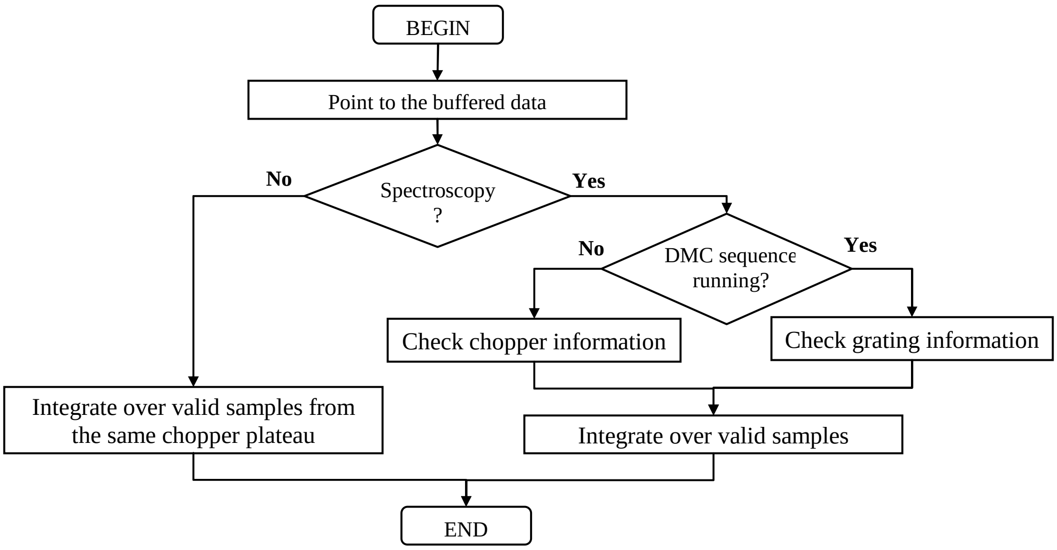


Figure 54. Flowchart of the Integration Procedure

5.9.5.6 Data

None

5.9.5.7 Resources

None

5.9.6 Temporal and Spatial Redundancy Reduction

5.9.6.1 Type

`T_S_Red.c` – procedure

5.9.6.2 Function

The aim of this procedure is to eliminate the temporal and spatial correlation of the sensor readings and resorting the data for better peroformance of lossless coding.

In case of Lossless Compression Mode, this module checks and selects the maximum data size allowed for lossless compression.

Maximum data size= 120 Kbytes:

- First 60 detectors (Figure 19) in spectroscopy are chosen
- First 256 detectors (Figure 20) in photometry are chosen for SWL SPU
- First 64 detectors (Figure 20) in photometry are chosen for LWL SPU

5.9.6.3 Interfaces

None

5.9.6.4 Dependencies

Called from Bol_ex, Bol3, Spec1_2 or Spec3 procedure.

5.9.6.5 Processing

See the flowchart in Figure 55 for spectroscopy and Figure 56 for photometry.

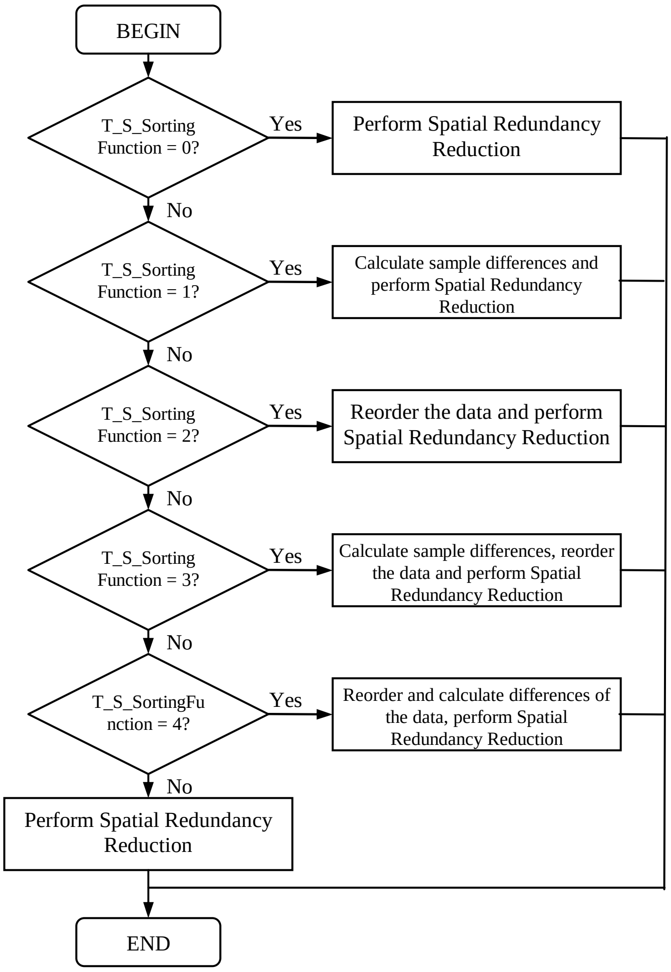


Figure 55. Flowchart of the Temporal and Spatial Redundancy Reduction Procedure for Spectroscopy

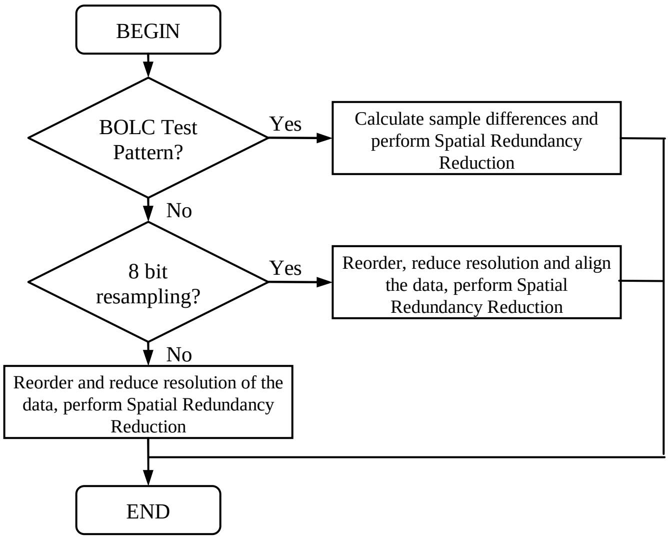


Figure 56. Flowchart of the Temporal and Spatial Redundancy Reduction Procedure for Photometry

5.9.6.6 Data

None

5.9.6.7 Resources

None

5.9.7 Lossless Coding

5.9.7.1 Type

`llc.c` – procedure

5.9.7.2 Function

This procedure performs the lossless coding of the data.

5.9.7.3 Interfaces

Call: pacs_srt.................................... Sorts the data for following compression algorithm

Call: initmodel (in pacs_cod.c).......... initialization of the dynamical model for the range coder

Call: pacs_encode (in pacs_cod.c) .... applies the model and calls the range coder

Call: deletemodel (in pacs_cod.c) ..... deletes the model

5.9.7.4 Dependencies

Called from Bol_ex, Bol3, Spec1_2 or Spec3 procedure.

5.9.7.5 Processing

See flowchart in Figure 57.

A library of Lossless Coding (LLC) algorithms is proposed in Appendix A.2.

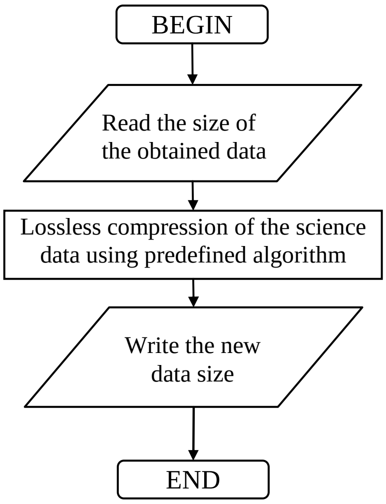


Figure 57. Flowchart of the Lossless Compression Procedure

The currently implemented Lossless Coding algorithm for science data can be found in Appendix A.1.2.

5.9.7.6 Data

None

5.9.7.7 Resources

None

5.9.8 Header Compression

5.9.8.1 Type

`dmch_cp.c` – procedure

5.9.8.2 Function

This procedure performs a lossless compression of the received DEC/MEC header.

5.9.8.3 Interfaces

Call: ReadInCircularBuffer .... Reads the DMC header from input buffer

5.9.8.4 Dependencies

Called from Bol_ex, Bol3, Bol4, Spec1_2, Spec3 or Spec4 procedure.

5.9.8.5 Processing

See flowchart in Figure 57.

A library of Lossless Coding (LLC) algorithms is proposed in Appendix A.2.

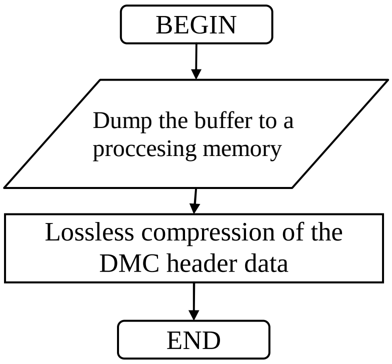


Figure 58. Flowchart of the DEC/MEC Header Compression Procedure

The currently implemented lossless compression algorithm for DEC/MEC header can be found in Appendix A.1.1.

5.9.8.6 Data

None

5.9.8.7 Resources

None

5.9.9 Fill_OutputBuffer

5.9.9.1 Type

`fill_out.c` – procedure

5.9.9.2 Function

The Fill_OutputBuffer procedure is used to prepare the compressed entity and build the PUS packets. It sets the packet ready-to-transmit to the data transmission function.

The compressed entity represents the data packet sent from the SPU to the DPU. It consists of the compressed science data and the compressed DEC/MEC header. It should be packaged and split into blocks, which fit in the telemetry packets (AD002) and made available to the DPU according to document RD007. The compressed entity packet structure is described in the document RD007.

The compressed entity is split by the SPU into blocks to fit into the telemetry packets (AD002). A header of 12 Bytes (Block Header) is added to each block before it is sent to the DPU (*Figure 59*). 8 Bytes of this header will be ignored in the DPU as described in RD007.

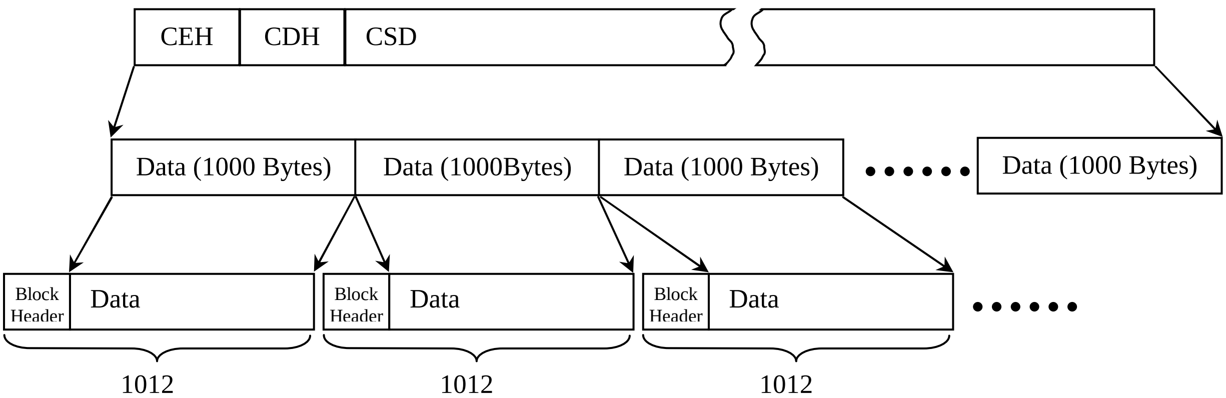


Figure 59. Splitting of the Compressed Entity and the Resulting Science Data Blocks

5.9.9.3 Interfaces

Call: KS_FIFOPut ....Sends the response packet to the FIFO (Data Transmission task)

Call: KS_TaskSleep ....Time delay

Call: WriteInCircularBuffer ...Store The TM packet in the output buffer

Call: KS_SemaSignal ....Signals the start of HK transmission

5.9.9.4 Dependencies

Called from Bol_ex, Bol3, Bol4, BUF_Bol, Spec1_2, Spec3, Spec4, Spec5 and BUF_Spec procedure.

5.9.9.5 Processing

See flowchart in Figure 60.

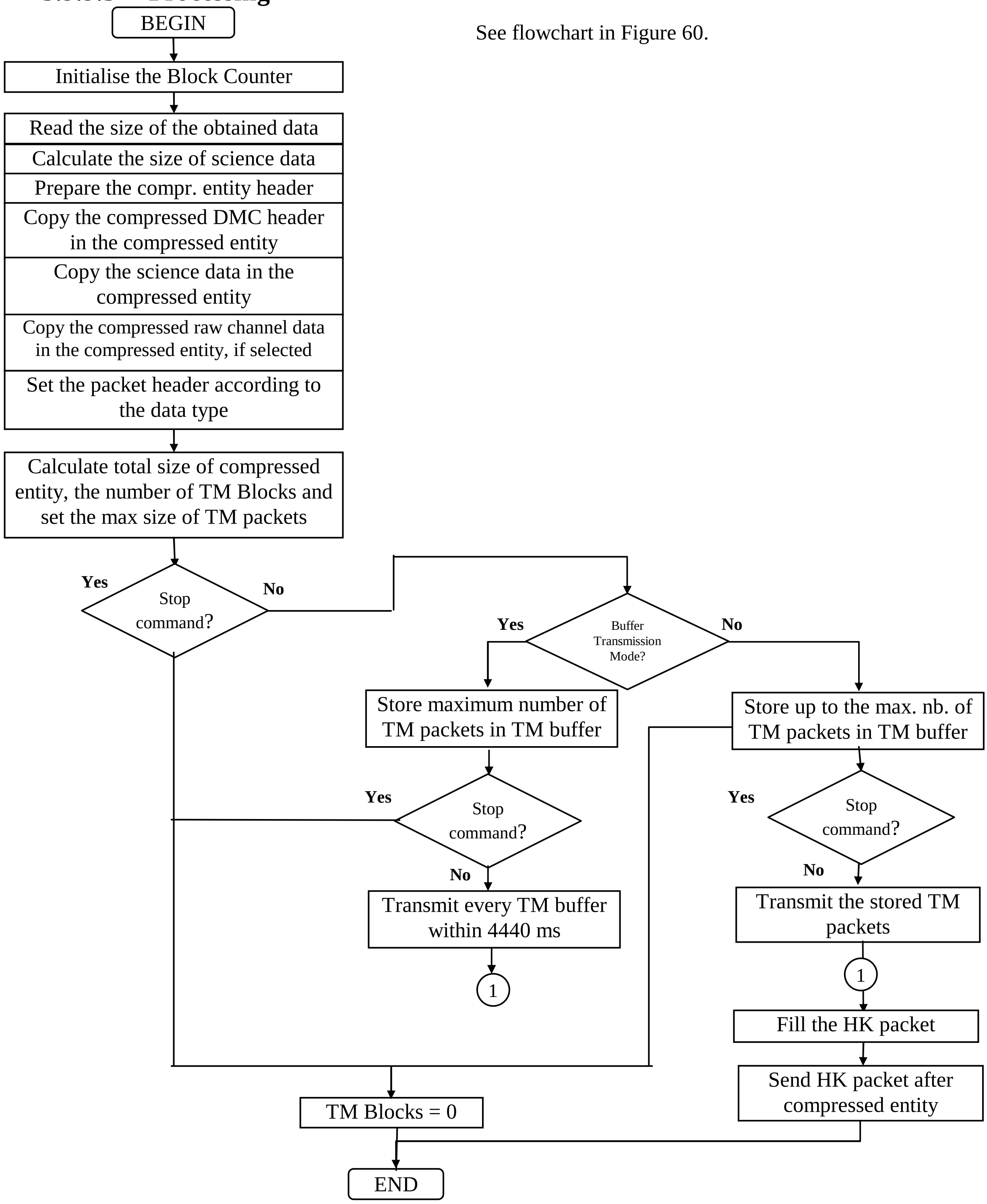


Figure 60. Flowchart of the Fill_OutputBuffer Procedure

5.9.9.6 Data

Writes data to the FIFO.

5.9.9.7 Resources

1 MB data memory

5.10 Supervisor for Test Mode

5.10.1 Type

`Spvs_Tst.c` – task

5.10.2 Function

The supervisor for test mode task listens to the data generators. It reads test data packets whenever received. If a start CSW command is signalled, then the test data packets are buffered and the compression is started.

It performs the supervisor activities while SPU test mode is activated.

5.10.3 Interfaces

Call: WriteInCircularBuffer ....Calls procedure to fill the Input Buffer with Science data

Call: KS_SemaSignal(SEMA_ASW)......Signals the start of the compression software task

5.10.4 Dependencies

Called from the main task.

5.10.5 Processing

See flowchart in Figure 61, Figure 62, Figure 63 and Figure 64.

‘`Compress_Now = true`’ in the flowchart means: “Close the actual buffer and make it ready for compression. Begin to store data in a new buffer.”

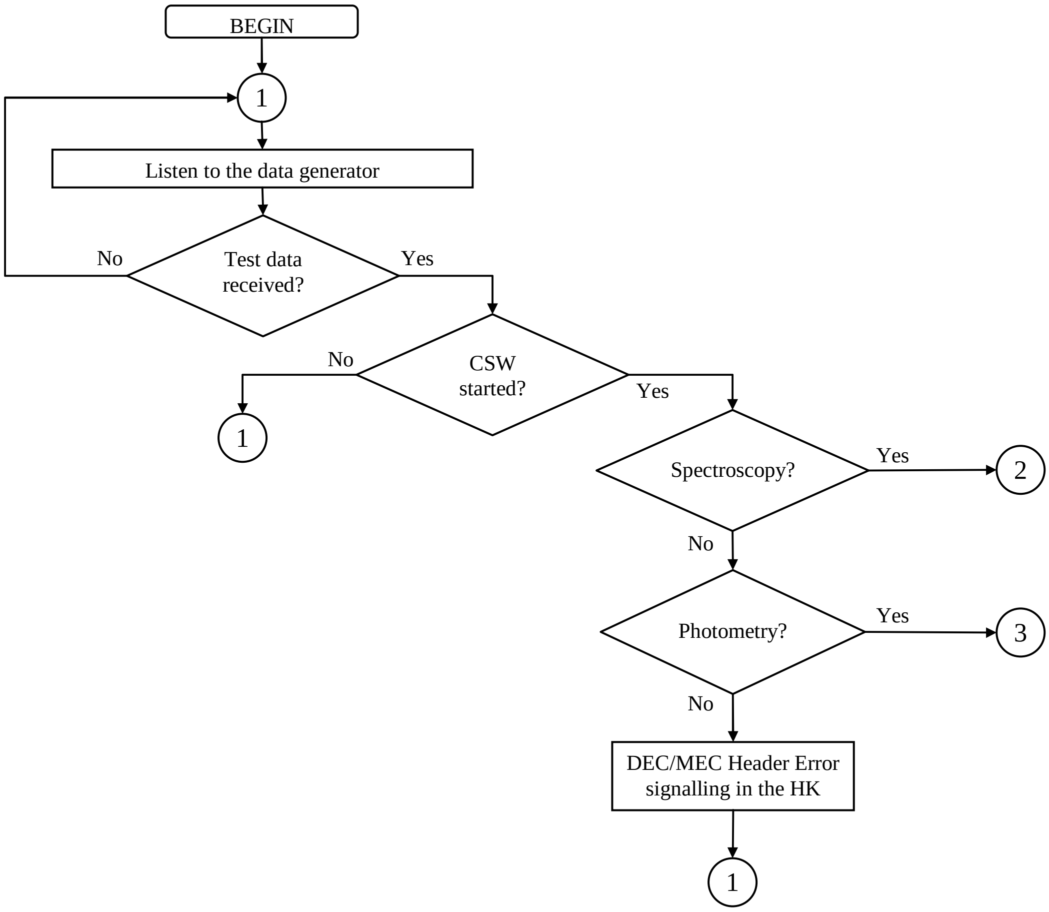
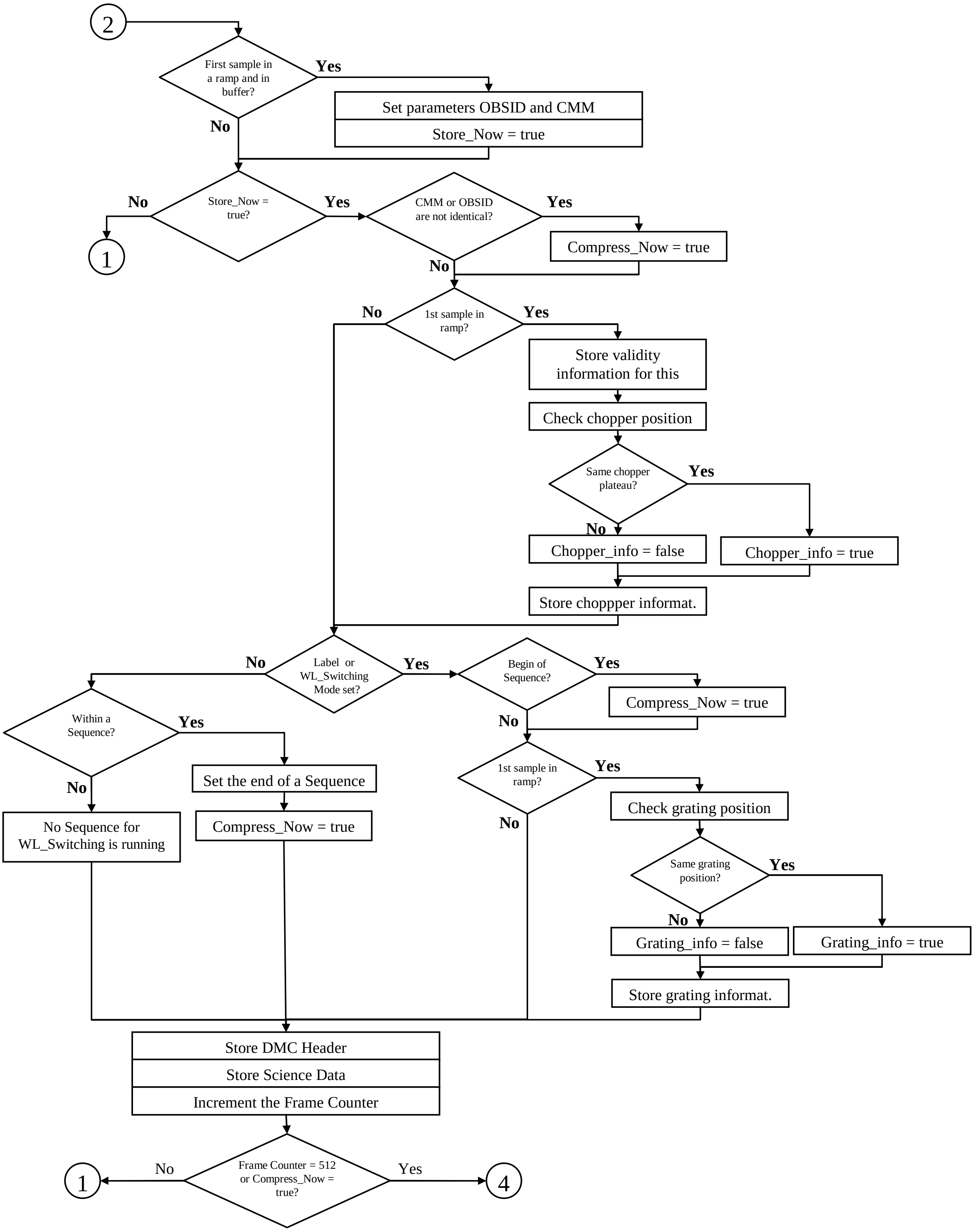


Figure 61. Flowchart of the Supervisor for Test Mode Task (Part 1)

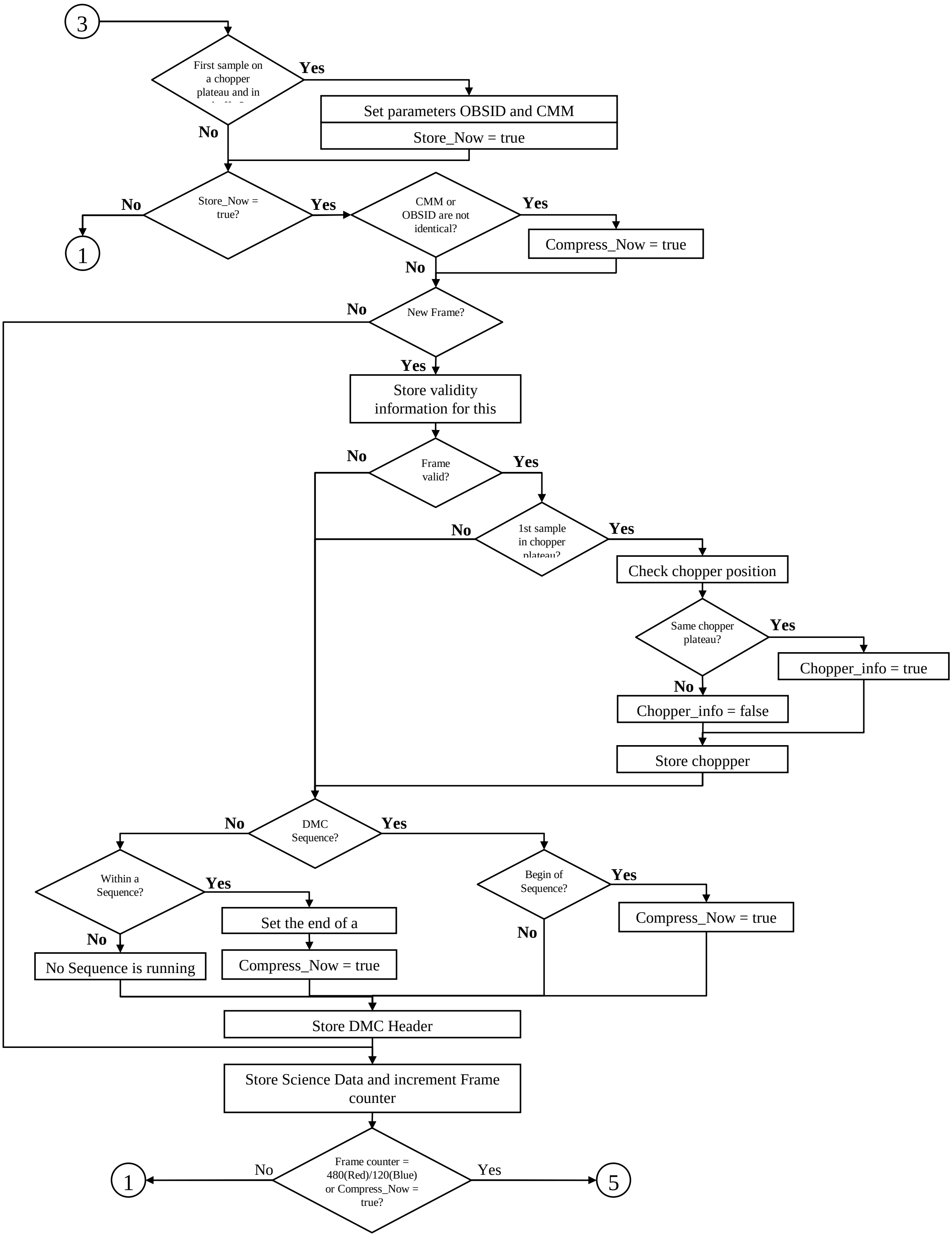
Figure 62. Flowchart of the Supervisor for Test Mode Task (Part 2)
See next page

Figure 63. Flowchart of the Supervisor for Test Mode Task (Part 3)
See page after next



2
First sample in a ramp and in buffer?
Yes
Set parameters OBSID and CMM
Store_Now = true
No
Store_Now = true?
No
1
Yes
CMM or OBSID are not identical?
Yes
Compress_Now = true
No
1st sample in ramp?
No
Yes
Store validity information for this
Check chopper position
Same chopper plateau?
Yes
No
Chopper_info = false
Chopper_info = true
Store choppper informat.
Label or WL_Switching Mode set?
No
Yes
Begin of Sequence?
Yes
Compress_Now = true
No
Within a Sequence?
Yes
Set the end of a Sequence
Compress_Now = true
No
No Sequence for WL_Switching is running
1st sample in ramp?
Yes
Check grating position
No
Same grating position?
Yes
No
Grating_info = false
Grating_info = true
Store grating informat.
Store DMC Header
Store Science Data
Increment the Frame Counter
Frame Counter = 512 or Compress_Now = true?
No
1
Yes
4

(Part 2)


3
First sample on a chopper plateau and in
Yes
No
Set parameters OBSID and CMM
Store_Now = true
No
Store_Now = true?
Yes
1
CMM or OBSID are not identical?
Yes
Compress_Now = true
No
No
New Frame?
Yes
Store validity information for this
No
Frame valid?
Yes
No
1st sample in chopper
Yes
Check chopper position
Same chopper plateau?
Yes
Chopper_info = true
No
Chopper_info = false
Store choppper
No
DMC Sequence?
Yes
Within a Sequence?
Yes
Set the end of a
No
No Sequence is running
Compress_Now = true
Begin of Sequence?
Yes
Compress_Now = true
No
Store DMC Header
Store Science Data and increment Frame counter
1
No
Frame counter = 480(Red)/120(Blue) or Compress_Now = true?
Yes
5

(Part 3)

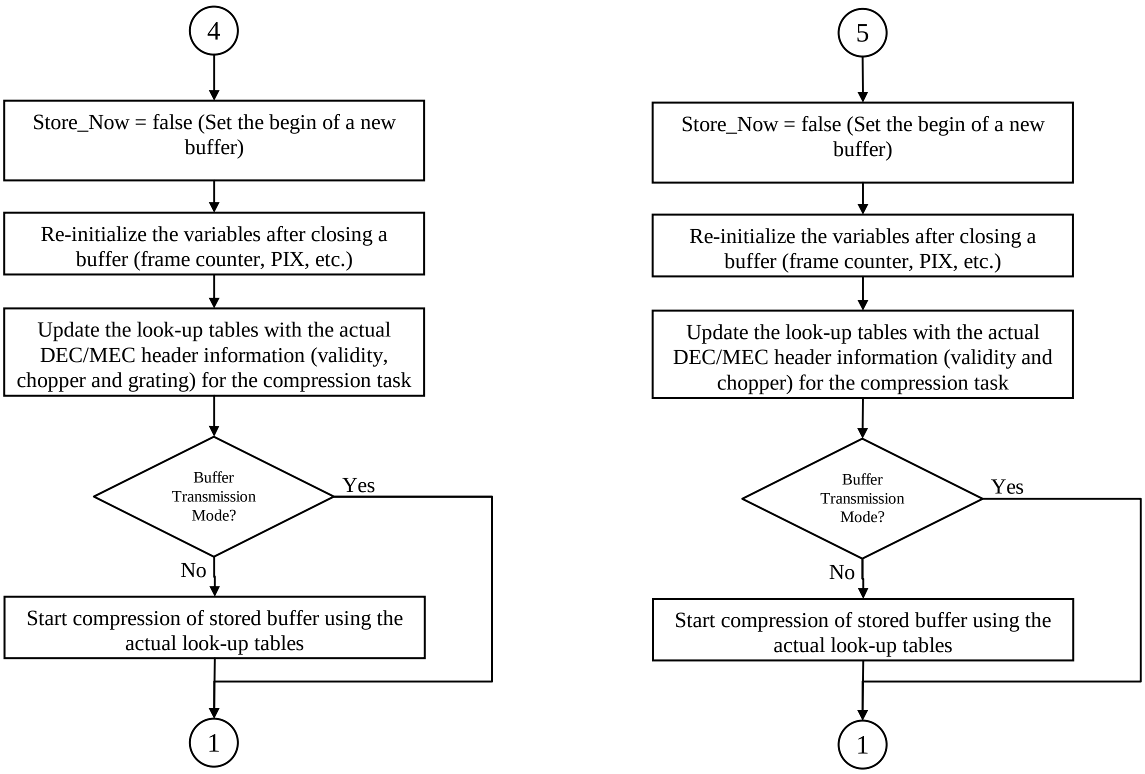


Figure 64. Flowchart of the Supervisor for Test Mode Task (Part 4)

5.10.6 Data

Read data from the data generator.

5.10.7 Resources

1 MB allocated Data Memory

5.11 Photometry Data Generator

5.11.1 Type

`phot_gen.c` – task

5.11.2 Function

This program generates simulated data for photometry.

5.11.3 Interfaces

Call: KS_SemaSignal(SPU_TEST).........Signals the start of the SPU Test Mode

Call: KS_TaskSleep ....Time Delay

5.11.4 Dependencies

Called from Spu_tst procedure.

5.11.5 Processing

See flowchart in Figure 65.

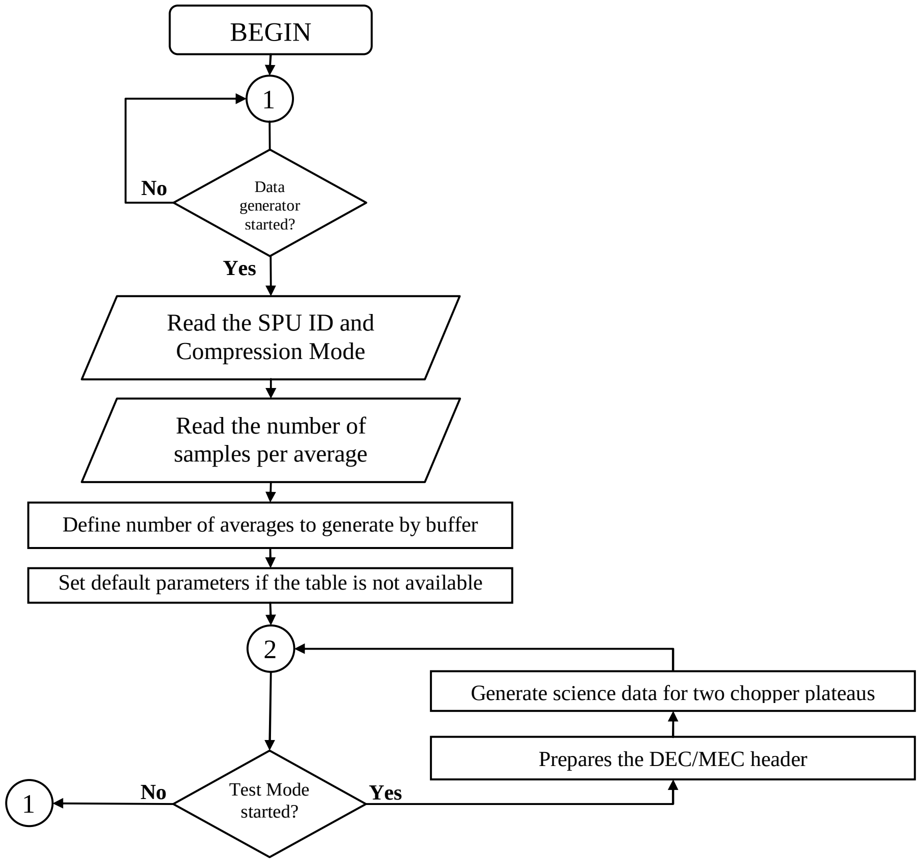


Figure 65. Flowchart of the Photometry Data Generator Task

5.11.6 Data

None

5.11.7 Resources

None

5.12 Spectroscopy Data Generator

5.12.1 Type

spec_gen.c – task

5.12.2 Function

This program generates simulated data for spectroscopy.

5.12.3 Interfaces

Call: KS_SemaSignal(SPU_TEST).........Signals the start of the SPU Test Mode.

Call: KS_TaskSleep ....Time Delay

5.12.4 Dependencies

Called from Spu_tst procedure.

5.12.5 Processing

See flowchart in Figure 66.

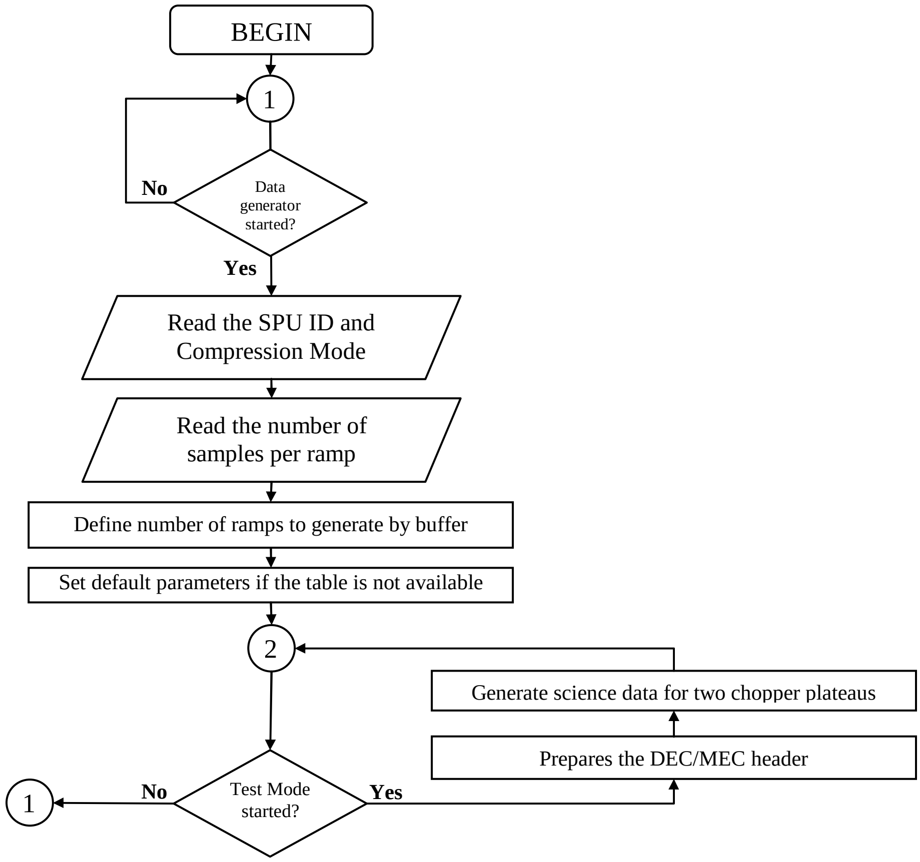


Figure 66. Flowchart of the Spectroscopy Data Generator Task

5.12.6 Data

None

5.12.7 Resources

None

6 Feasibility and Resource Estimates

6.1 Memory Allocation

The operations of the SPU HLSW are limited by the available memory size. Figure 67 represents the memory allocation within one SPU board. Table 7 shows the memory allocation for the SPU HLSW.

Memory Type	Memory Size [MB]	Memory Allocation [%]	Description
EEPROM	1.5	17	for storage and backup of the program code to avoid several uplinks of the same program code in case of SPU switch off/on, RAM corruption etc.
Program Memory (PRAM)	3.0	25[3]	used to store and run the program code.
Data Memory (DRAM)	4.0	94	for static data allocation and data processing storage.
SMCS Memory (DPRAM)	32 kB	87.5	used for data transfer between SPU – DEC/MEC and SPU – DPU.

Table 7. Memory Allocation of the SPU HLSW.

The memory usage is at least 10% less than the memory allocated.

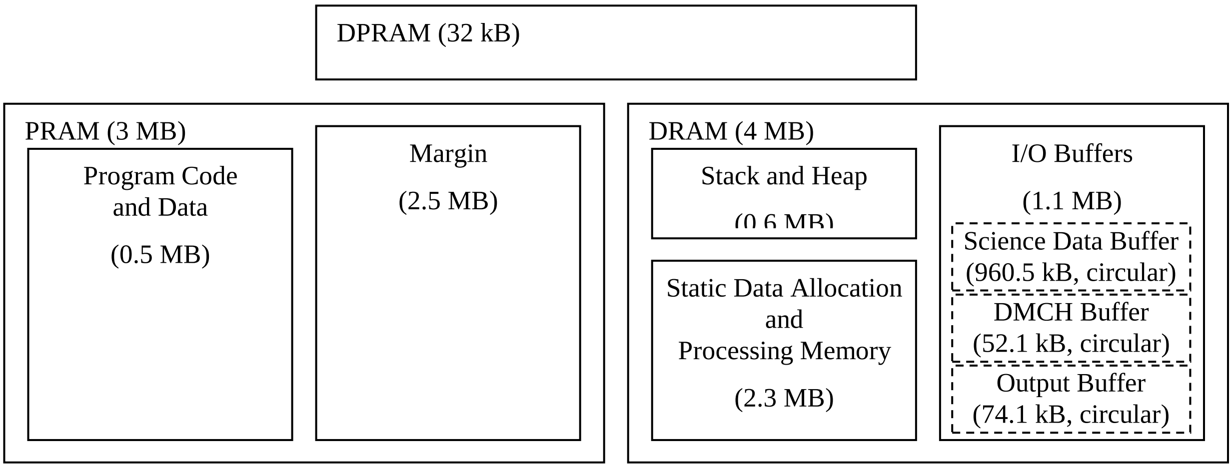


Figure 67. SPU HLSW Memory Map.

[3] In the buffer transmission mode (see Section 5.8.5 and Section 5.8.9) program memory is also used for data buffering. Therefore the program memory allocation increases to 99.95%.

In data memory (DRAM) three circular buffers are available per SPU board for data acquisition, storage and transmission, in addition to the processing memory. For data compression the SPU HLSW swaps between two 0.5 Mbytes buffered data. After the buffering of 0.5 Mbytes of input data (science data and DEC/MEC header) the compression of this data starts while the second 0.5 Mbytes buffer is filled.

The memory map is given in RD016. The PRAM is not protected from overwriting by the HLSW because the PRAM is used by two compression modes for data buffering (Buffer Transmission Mode and 4s Reset mode in spectroscopy). Furthermore, the write command is able to access the PRAM area.

6.2 Processing Power

The SPU consists of two CPU boards (TSC21020). The clock frequency of each CPU is 18 MHz with 54 MFLOPs (peak) and 36 MFLOPs (sustained) (see RD012, page 11). The exact CPU workload derived for all compression modes, for a HLSW version, can be found in the respective HLSW user manual (RD016).

6.3 Application SW Estimates

In this section, additional requirements of the compression SW (ASW) on the signal in order to achieve the required compression ratio are given.

6.3.1 Compression Ratio

Nominal PACS science data rate is about 120 kbits/s in prime mode. Therefore, the allocated rate for science data excluding the header is about 118 kbits/s.

Based on this assumption the minimum required compression ratio is calculated below.

Photometry

Science Data Input	(2048 + 512) detectors x 40 Hz x 16 bit = 1600 kbit/s
Minimum CR in photometry:	1600/118 = **13.56**

Spectroscopy

Science Data Input	2 (boards) x (26x18) detectors x 256 Hz x 16 bit = 3744 kbit/s
Minimum CR in spectroscopy	3744/118 = **31.73**

6.3.2 Entropy Estimates for Redundancy Reduction

In this Section, we calculate the redundancy in the signal, in terms of bits, that is required to achieve the minimum compression ratio. We only consider the lossless compression part of the ASW. A compression factor (see Table 8) should be already achieved by the lossy compression module in nominal compression modes.

In the lossless compression module one reference value is kept every t_{ref} time interval. The differences to the reference are calculated (see also Figure 68) and coded in nbit bits (noise). For more details see Section 5.9.6.

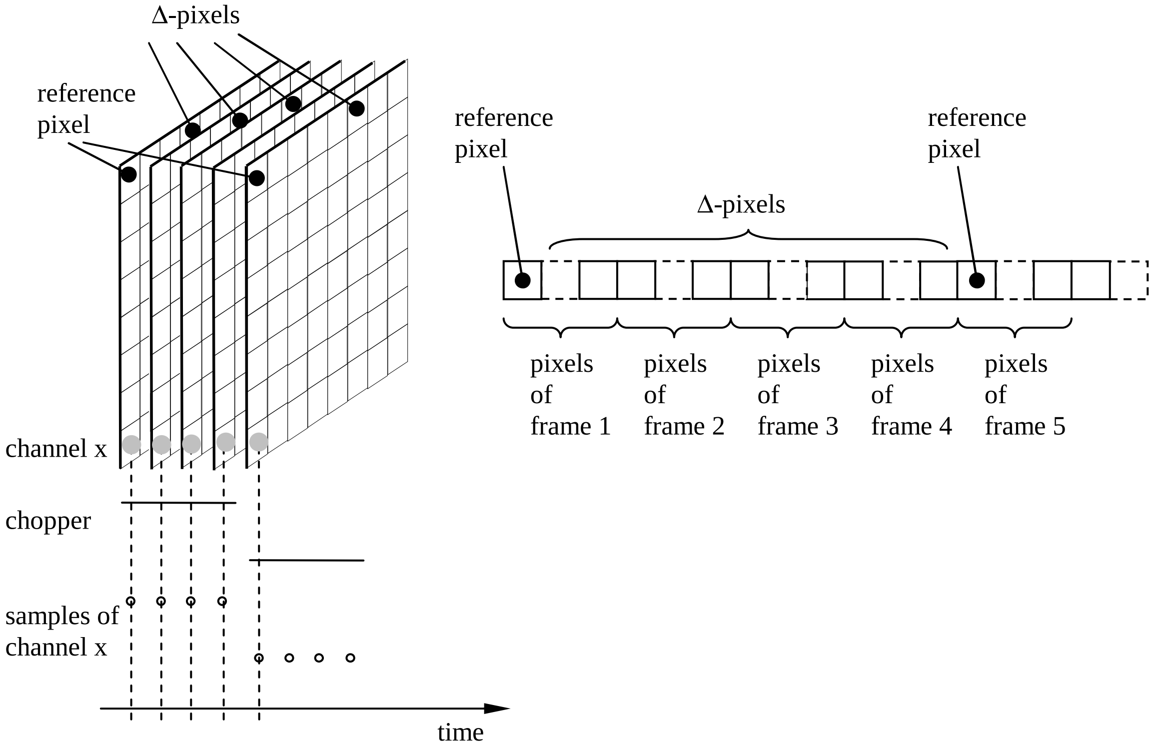


Figure 68. Redundancy Reduction.

The compression ratio (CR) achieved by the redundancy reduction module can be calculated as follows:

$$CR = [16 \text{ x } N_d \text{ x } t_{ref} \text{ x } \nu_S / CR\_fit] / [chop\_pos \text{ x } 16 + nbit \text{ x } (N_d \text{ x } t_{ref} \text{ x } \nu_S / CR\_fit - chop\_pos)]$$

where

N_d .... number of detectors
ν_S .... sampling frequency
CR_fit.......... compression ratio of the previous lossy compression step (averaging/ramp fitting module)
chop_pos..... number of chopper positions (one reference per chopper plateau).

Therefore, the maximum number of bits allocated to the noise can be calculated as follows:

$$nbit = [(16 \times N_d \times t_{ref} \times \nu_S / CR\_fit) / CR - chop\_pos \times 16] / [N_d \times t_{ref} \times \nu_S / CR\_fit - chop\_pos] .$$

Table 8 depicts the maximum number of bits allocated to the noise required to fulfil the TM rate. They are calculated for 4 Hz chopper frequency.

Observing Mode	Minimum CR	CR by lossy compression module	CR by lossless compression module	number of bits allocated to the noise
Photometry	13.56	4	3.390	**4.72**
Spectroscopy	31.73	32	0.992	**16.13**
		16	1.983	**8.06**
		8	3.966	**4.03**
		4	7.933	**2.02**
		2	15.865	**1.01**

Table 8. Number of Bits Allocated to the Noise.

6.4 Data Rates Estimate

Table 9 and Table 10 show the data rates estimate for all compression modes in photometry and spectroscopy. These numbers are calculated as presented in the next subsections.

Compression Mode in Photometry	Description	Selected Detectors		Expected TM Rate [kbits/s]			Packets per second
		SPU LWL	SPU SWL	SPU LWL	SPU SWL	Total	
BOL0	Half Compression Mode	512	2048	87.44	29.93	**117.37**	15.05
BOL1	Default Mode	512	2048	81.95	21.53	**103.48**	13.17
BOL2	Double Compression Mode	512	2048	41.68	11.45	**53.13**	6.76
BOL3	Lossless Compression Mode	512	2048	323.39	81.89	**405.28**	51.57
		148	592	94.53	24.67	**119.20**	15.17
BOL4	Transparent Mode	512	2048	1289.11	323.28	**1612.39**	205.15
		37	148	94.48	24.63	**119.11**	15.16
BUF_Bol	Buffer Transmission Mode	512	2048	94.50	118.60	**213.10**	27.11

Table 9. Expected Data Rates for the Compression Modes in Photometry

Compression Mode in Spectroscopy	Description	Selected Detectors		Expected TM Rate [kbits/s]			Packets per second
		SPU LWL	SPU SWL	SPU LWL	SPU SWL	Total	
SPEC1	Default Mode	450	450	58.76	58.76	**117.52**	14.95
SPEC2	Double Compression Mode	450	450	30.47	30.47	**60.94**	7.75
SPEC3	Lossless Compression Mode	450 57	450 57	454.88 59.53	454.88 59.53	**909.75** **119.06**	115.75 15.14
SPEC4	Transparent Mode	450 14	450 14	1812.94 58.46	1812.94 58.46	**3625.88** **116.92**	461.34 14.88
SPEC5	Compression Mode for 4s Reset	450	450	58.73	58.73	**117.47**	14.95
BUF_Spec	Buffer Transmission Mode	450	450	122.71	122.71	**245.42**	31.23

Table 10. Expected Data Rates for the Compression Modes in Spectroscopy

The data rate for one raw channel in photometry is

40 Hz x 16 bit = **0,625 kbits/s**

and for one raw channel in spectroscopy

256 Hz x 16 bit = **4 kbits/s** .

6.4.1 Photometry

The nominal data rates in photometry are calculated as follows:

Step 1: Averaging

Since we have to average 4 samples, we reduce the input science data by a factor of 4.

The input science data of 1280 + 320 = 1600 kbit/s are reduced to 320 + 80 = 400 kbit/s here.

Step 2: Redundancy Reduction

- **SWL**:

  16 bits x 10 (averages) x 2048 (detectors) after data averaging = 327680 bits per array

  (4 x 16 bits) + 4 bits x (10 x 2048 detectors - 4) = 81968 bits per array

  The ratio achieved in this step is 327680/81968 = 3.9977

- **LWL**:

  16 bits x 10 (averages) x 512 (detectors) after data averaging = 81920 bits per array

(4 x 16 bits) + 4 bits x (10 x 512 detectors - 4) = 20528 bits per array

The ratio achieved for the red SPU is 81920/20528 = 3.9906

The science data is reduced to ***80.0469*** *+* ***20.0469*** *= 100.0938 kbit/s here.*

Table 11 shows the nominal data rates for photometry. The calculation of the header rates is described below.

	SWL [kbits/s]	**LWL [kbits/s]**	**SWL & SWL [kbits/s]**
Compressed Science Data	80.0469	20.0469	100.0938
Compressed DEC/MEC Header Rate	1.3333	1.3333	2.6667
Compressed Entity Header	0.0729	0.0182	0.0911
Packet Header	0.5000	0.1289	0.6289
Total	**81.9531**	**21.5273**	**103.4805**

Table 11. Nominal Data Rates for Photometry.

Compressed DEC/MEC Header Rate

If we consider 512 Bytes per compressed entity for the compressed DEC/MEC header then we get

SWL: 512 Bytes / 3 s = 170.67 Bytes/s (**1.333 kbits/s**)

LWL: 2048 Bytes / 12 s = 170.67 Bytes/s (**1.333 kbits/s**)

Compressed Entity Header Rate

If we consider 28 Bytes per compressed entity for the CEH then we get

SWL: 28 Bytes / 3 s = 9.333 Bytes/s (**0.0729 kbits/s**)

LWL: 28 Bytes / 12 s = 2.333 Bytes/s (**0.0182 kbits/s**)

Packet Header Rate

If we consider a packet size of 1000 Bytes we get the number of packets

SWL (28B + 512B + [3s x 2048 x 40Hz x 2B / 15.9976(CR)]) / 1000B = 31.265

→ 32 packets/buffer; 32 packets / 3s = 10.667 packets/s

LWL (28B + 2048B + [12s x 512 x 40Hz x 2B / 15.9908 (CR)]) / 1000B = 32.814

→ 33 packets/buffer; 33 packets / 12s = 2.75 packets/s

If we consider 6 Bytes per packet then we get a packet header rate of

SWL: 10.667 packets/s x 0.046875 kbits = **0.5 kbits/s**

LWL: 2.75 packets/s x 0.046875 kbits = **0.1289 kbits/s**

6.4.2 Spectroscopy

The nomunal data rates in spectroscopy are calculated as follows:

Step 1: Ramp Fitting

Since we have to fit over 8 samples, we reduce the input science data by a factor of 8.

The input science data of 2 x 1800 = 3600 kbit/s are reduced to 2 x 225 = 450 kbit/s here.

Step 2: Redundancy Reduction

- **SWL & LWL**:

  16 bits x 32 (slopes) x 450 (detectors) after data fitting = 230400 bits per array

  (4 x 16 bits) + 4 bits x (32 x 450 detectors - 4) = 57648 bits per array

  The ratio achieved in this step is 230400/57648 = 3.9967

*The science data is reduced to 2 x **56.2969** = 112.5938 kbit/s here.*

Table 12 shows the nominal data rates for spectroscopy. The calculation of the header rates is described below.

	SWL [kbits/s]	LWL [kbits/s]	SWL & SWL [kbits/s]
Compressed Science Data	56.2969	56.2969	112.5938
Compressed DEC/MEC Header Rate	2.0000	2.0000	4.0000
Compressed Entity Header	0.1094	0.1094	0.2188
Packet Header	0.3516	0.3516	0.7031
Total	**58.7578**	**58.7578**	**117.5156**

Table 12. Nominal Data Rates for Spectroscopy.

Compressed DEC/MEC Header Rate

If we consider 512 Bytes per compressed entity for the compressed DEC/MEC header then we get

SWL & LWL: 512 Bytes / 2 s = 256 Bytes/s (**2 kbits/s**)

Compressed Entity Header Rate

If we consider 28 Bytes per compressed entity for the CEH then we get

SWL & LWL: 28 Bytes / 2 s = 14 Bytes/s (**0.1094 kbits/s**)

Packet Header Rate

If we consider a packet size of 1000 Bytes we get the number of packets

SWL & LWL: (28B + 512B + [2s x 450 x 256Hz x 2B / 31.97 (CR)]) / 1000B = 14.95

→ 15 packets/buffer; 15 packets / 2s = 7.5 packets/s

If we consider 6 Bytes per packet then we get a packet header rate of

SWL & LWL: 7.5 packets/s x 0.046875 kbits = **0,3516 kbits/s**

7 User Requirements vs. Software Traceability Matrix

User Requirements	Software Requirements	Description
SPU-OBS-ON1	**SPU-SUSW-2610**	SPU handover to HLSW (see RD015 page 41)
SPU-OBS-ON2	**SPU-HSR-FN0200**	HK to DPU
SPU-OBS-OF1	---	No data storage before the switch off of the SPU unit
SPU-OBS-SW1	**SPU-HSR-FN0102..105**	HLSW performs the data reduction and compression according to DEC/MEC header
SPU-OBS-SW2	**SPU-HSR-FN0005** **SPU-HSR-FN0202** **SPU-HSR-FN0203**	HLSW has the possibility to detect data corruption in the Loaded memory area
SPU-OBS-SW3	**SPU-HSR-OP0611**	The compression results are included in the HK whenever compression is started
SPU-OBS-SW4	**SPU-HSR-FN0020..30**	HLSW upgrades the parameters for data reduction
SPU-OBS-SW5	**SPU-HSR-FN0014,** **SPU-HSR-IF0003,** **SPU-HSR-IF0070**	HLSW aborts data processing whenever a stop command is received
SPU-OBS-SW6	**SPU-HSR-FN0015,** **SPU-HSR-IF0003,** **SPU-HSR-IF0080**	HLSW starts data processing whenever a start command is received
SPU-OBS-SW7	**SPU-HSR-FN0012,** **SPU-HSR-IF0003,** **SPU-HSR-IF0050**	HLSW performs the warm reset when a reset command is received
SPU-OBS-SW8 deleted	**SPU-HSR-FN0018,** **SPU-HSR-IF0003,** **SPU-HSR-IF0040**	HLSW starts the peak-up program when a peak-up command is received
SPU-OBS-SW9	**SPU-HSR-OP0400,** **SPU-HSR-OP0500**	HLSW supports all the operating and observing modes of the PACS instrument
SPU-OBS-SW10	**SPU-HSR-FN0016,** **SPU-HSR-FN0017,** **SPU-HSR-IF0003,** **SPU-HSR-IF0090,** **SPU-HSR-IF0100,** **SPU-HSR-OP0640**	HLSW generates simulated data for test purpose, triggered by DPU command
SPU-OBS-SW11	**SPU-HSR-FN0007,** **SPU-HSR-FN0008,** **SPU-HSR-IF0002,** **SPU-HSR-IF0003,** **SPU-HSR-IF0020**	HLSW could dump data from RAM
SPU-OBS-SW12	**SPU-HSR-FN0004,** **SPU-HSR-FN0005,** **SPU-HSR-FN0006,** **SPU-HSR-IF0003,** **SPU-HSR-IF0010**	HLSW could Load data into RAM
SPU-OBS-SW13	**SPU-HSR-FN0009,** **SPU-HSR-FN0010,**	HLSW could check a specific RAM area

	SPU-HSR-IF0002, SPU-HSR-IF0003, SPU-HSR-IF0030,	
SPU-OBS-SW14	**SPU-HSR-FN0020..30, SPU-HSR-IF0003, SPU-HSR-IF0120**	HLSW could write detector selection tables and detector constant tables to RAM
SPU-OBS-SW15 deleted	**SPU-HSR-FN0019, SPU-HSR-IF0003, SPU-HSR-IF0110**	HLSW performs the Bolometer Background Cancelling mode whenever a BBC command is received
SPU-OBS-CR1	**SPU-HSR-FN0102..105**	HLSW performs the data reduction and compression according to DEC/MEC header
SPU-OBS-CR2	**SPU-HSR-FN0300**	No data loss during data acquisition
SPU-OBS-CR3	**SPU-HSR-OP0600..04, SPU-HSR-OP0610..11, SPU-HSR-OP0620, SPU-HSR-OP0625, SPU-HSR-OP0630, SPU-HSR-OP0631, SPU-HSR-OP0632, SPU-HSR-OP0640, SPU-HSR-OP0650, SPU-HSR-OP0660..62, SPU-HSR-OP0670**	HLSW supports several compression modes
SPU-OBS-CR4	**SPU-HSR-IF0200..01**	Compression data is split into packets which fit into the telemetry packet
SPU-OBS-CR5	**SPU-HSR-IF0200..01**	Compressed data packet structure fulfil the protocol described in RD007
SPU-OBS-CR6	**SPU-HSR-IF0300..04**	HLSW is able to receive data packets from DEC/MEC
SPU-OBS-CR7	**SPU-HSR-RS0001, SPU-HSR-RS0100**	HLSW is adapted to the SPU hardware
SPU-OBS-CR8	**SPU-HSR-IF0003**	HLSW acknowledges the reception of all DPU commands
SPU-OBS-CR9	**SPU-HSR-FN0200**	HLSW sends HK to DPU
SPU-OBS-CR10	**SPU-HSR-FN0200**	Regular rate for the HLSW HK transmission to DPU
SPU-OBS-CR11	**SPU-HSR-FN0201**	“Are You Alive” SPU is available in the HLSW HK
SPU-OBS-TM1	**SPU-HSR-OP0610**	HLSW supports the PACS prime mode (120 Kbits/s)
SPU-OBS-TM2	**SPU-HSR-OP0660**	HLSW supports the PACS burst mode (~300 Kbits/s)
SPU-OBS-TM3	**SPU-HSR-OP0620**	HLSW supports the PACS parallel mode
SPU-OBS-TM4	**SPU-HSR-OP0611**	The compression results are included in the HK whenever compression is started
SPU-OBS-TM5	**SPU-HSR-IF0200..01**	Compression data is split into packets which fit into the telemetry packet

8 Software Requirements vs. Components Traceability Matrix

Software Requirements	Component	Description
SPU-HSR-FN0001	`spu_io.c, watchp.c`	HLSW is able to receive command packets from DPU
SPU-HSR-FN0002	`watchp.c`	HLSW is able to identify the DPU command
SPU-HSR-FN0003	`watchp.c`	HLSW aborts data compression if DPU command is received
SPU-HSR-FN0004	`watchp.c, Load.c`	HLSW can load data into SPU RAM
SPU-HSR-FN0005	`Load.c`	HLSW checks the consistency of the data before the load
SPU-HSR-FN0006 transfered to **SPU-HSR-IF0011**	`Load.c`	Data load only to DRAM, Ext. DRAM and DPRAM
SPU-HSR-FN0007	`watchp.c, dump.c`	HLSW can dump data from SPU RAM
SPU-HSR-FN0008 transfered to **SPU-HSR-IF0021**	`dump.c`	Data dump only from PRAM, DRAM, Ext. DRAM and DPRAM
SPU-HSR-FN0009	`watchp.c, check.c`	HLSW can check SPU RAM
SPU-HSR-FN0010	`check.c`	Data check only for PRAM, DRAM, Ext. DRAM and DPRAM
SPU-HSR-FN0011	`watchp.c, perform.c`	HLSW checks consistency of the parameter of the perform activity command
SPU-HSR-FN0012	`w_reset.c, fill_in.c, spu_io.c`	HLSW can perform warm reset
SPU-HSR-FN0013	`Rc_Sel.c`	HLSW can transmit data from selected channels in raw
SPU-HSR-FN0014	`Str_Stp.c`	HLSW can abort data compression
SPU-HSR-FN0015	`Str_Stp.c`	HLSW can start data compression
SPU-HSR-FN0016	`spu_tst.c, phot_gn.c, Spvs_Tst.c`	HLSW can generate simulated photometry data on board for test purposes
SPU-HSR-FN0017	`spu_tst.c, spec_gn.c, Spvs_Tst.c`	HLSW can generate simulated spectroscopy data on board for test purposes
SPU-HSR-FN0018 deleted	`peak_up.c, pkup_pg.c, csw.c`	HLSW provides an interface to the peak-up program
SPU-HSR-FN0019 deleted	`bol_bc.c, bbc_pg.c, csw.c`	HLSW provides an interface to the bolometer background cancelling program
SPU-HSR-FN0020	`watchp.c, write.c`	HLSW checks consistency of the parameter of the write command
SPU-HSR-FN0021	`write.c`	HLSW can load detectors table from subarray 1 in photometry
SPU-HSR-FN0022	`write.c`	HLSW can load detectors table from subarray 2 in photometry
SPU-HSR-FN0023	`write.c`	HLSW can load detectors table from subarray 3 in photometry
SPU-HSR-FN0024	`write.c`	HLSW can load detectors table from subarray 4 in photometry
SPU-HSR-FN0025	`write.c`	HLSW can load detectors table from subarray 5 in

		photometry
SPU-HSR-FN0026	`write.c`	HLSW can load detectors table from subarray 1 in spectroscopy
SPU-HSR-FN0027	`write.c`	HLSW can load detectors table from subarray 2 in spectroscopy
SPU-HSR-FN0028	`write.c`	HLSW can load detector constants table in photometry
SPU-HSR-FN0029	`write.c`	HLSW can load detector constants table in spectroscopy
SPU-HSR-FN0030	`write.c`	HLSW can load simulated data table for SPU test mode
SPU-HSR-FN0031	`C2EEPROM.c`	HLSW can copy SW code from RAM to EEPROM
SPU-HSR-FN0032	`DMC_Con.c, spu_io.c`	HLSW connects to DEC/MEC upon DPU command
SPU-HSR-FN0100	`spu_io.c, supervs.c`	HLSW is able to receive data packets from DPU
SPU-HSR-FN0101	`csw.c`	HLSW can start data compression
SPU-HSR-FN0102	`Supervs.c`	HLSW can distinguish between photometry and spectroscopy data
SPU-HSR-FN0103	`Supervs.c, fill_in.c`	HLSW is able to buffer DEC/MEC data to memory
SPU-HSR-FN0104	`Supervs.c, csw.c`	For real time processing procedures are implemented in HLSW to handle the buffers
SPU-HSR-FN0105	`csw.c, bol_ex.c, bol3.c, bol4.c, buf_bol.c, spec1_2.c, spec3.c, spec4.c, spec5.c, buf_spec.c`	HLSW supports eleven compression modes
SPU-HSR-FN0200	`hk.c, datatx.c`	HLSW send permanently HK packets to DPU
SPU-HSR-FN0201	`hk.c`	„Are You Alive“ SPU is implemented in the HLSW HK
SPU-HSR-FN0202	`hk.c`	EDAC check status is implemented in the HLSW HK
SPU-HSR-FN0203	`hk.c`	Ground people are responsible for analyzing the failing memory addresses detected using EDAC check
SPU-HSR-FN0300	`All`	HLSW fulfils real time data processing requirement
SPU-HSR-FN0301	`Supervs.c, integ.c`	HLSW doesn't allow science data loss
SPU-HSR-IF0001	`watchp.c`	HLSW aborts any running activity, if DPU command is received
SPU-HSR-IF0002	`watchp.c, Load.c, dump.c, C2EEPROM.c, check.c, DMC_Con.c, perform.c, w_reset.c, Rc_Sel.c, Str_Stp.c, spu_tst.c, write.c, datatx.c`	Maximum response time for DPU commands is 200ms
SPU-HSR-IF0003	`watchp.c, Load.c, dump.c, C2EEPROM.c, check.c, DMC_Con.c, perform.c, w_reset.c, Rc_Sel.c, Str_Stp.c, spu_tst.c, write.c, datatx.c`	HLSW acknowledges any received DPU command according to the defined protocol
SPU-HSR-IF0010	`watchp.c, Load.c, datatx.c`	HLSW acknowledges the execution of the load command
SPU-HSR-IF0011	`Load.c`	Data load only to DRAM, Ext. DRAM and DPRAM
SPU-HSR-IF0020	`watchp.c, dump.c, datatx.c`	HLSW acknowledges the execution of the dump command
SPU-HSR-IF0021	`dump.c`	Data dump only from PRAM, DRAM, Ext. DRAM and DPRAM

SPU-HSR-IF0030	`watchp.c, check.c, datatx.c`	HLSW acknowledges the execution of the check command
SPU-HSR-IF0040 deleted	`watchp.c, perform.c, peak_up.c, pkup_pg.c, csw.c, datatx.c`	HLSW acknowledges the reception of the peak-up command
SPU-HSR-IF0050	`watchp.c, perform.c, w_reset.c, datatx.c, spu_io.c`	HLSW acknowledges the reception of the warm reset command
SPU-HSR-IF0060	`watchp.c, perform.c, Rc_Sel.c, datatx.c`	HLSW acknowledges the execution of the raw channel selection command
SPU-HSR-IF0070	`watchp.c, perform.c, Str_Stp.c, datatx.c`	HLSW acknowledges the reception of the stop command
SPU-HSR-IF0080	`watchp.c, perform.c, Str_Stp.c, datatx.c`	HLSW acknowledges the reception of the start command
SPU-HSR-IF0090	`watchp.c, perform.c, spu_tst.c, phot_gn.c, Spvs_Tst.c, datatx.c`	HLSW acknowledges the reception of the SPU test in photometry command
SPU-HSR-IF0100	`watchp.c, perform.c, spu_tst.c, spec_gn.c, Spvs_Tst.c, datatx.c`	HLSW acknowledges the reception of the SPU test in spectroscopy command
SPU-HSR-IF0110 deleted	`watchp.c, perform.c, bol_bc.c, bbc_pg.c, csw.c, datatx.c`	HLSW acknowledges the reception of the Bolometer Background Cancelling command
SPU-HSR-IF0120	`watchp.c, write.c, datatx.c`	HLSW acknowledges the execution of the write command
SPU-HSR-IF0130	`watchp.c, perform.c, C2EEPROM.c, datatx.c`	HLSW acknowledges the reception of the Copy Data to EEPROM command
SPU-HSR-IF0131	`C2EEPROM.c`	No HK packet are transmitted to DPU while data are copied to EEPROM
SPU-HSR-IF0140	`watchp.c, perform.c, DMC_Con.c, spu_io.c`	HLSW acknowledges the reception of the Connect to DEC/MEC command
SPU-HSR-IF0200	`fill_out.c, datatx.c`	HLSW is able to transmit TM packets to DPU
SPU-HSR-IF0201	`fill_in.c, datatx.c, spu_io.c`	HLSW can buffer a maximum number of 75 TM packets
SPU-HSR-IF0300	`Supervs.c`	HLSW doesn't acknowledge the reception of DEC/MEC packets
SPU-HSR-IF0301	`Supervs.c, csw.c, hk.c`	HLSW can detect some corrupted fields in the DEC/MEC packets
SPU-HSR-IF0302	`Supervs.c`	HLSW can distinguish between photometry and spectroscopy packets
SPU-HSR-IF0303	`spu_io.c, hk.c`	DEC/MEC link connection status is set in the HLSW HK
SPU-HSR-IF0304	`Supervs.c, csw.c, hk.c`	DEC/MEC packet size
SPU-HSR-OP0001 obsolete	`Supervs.c, csw.c`	HLSW set the buffer granularity in spectroscopy to 8 Hz
SPU-HSR-OP0002	`Supervs.c, fill_in.c, csw.c`	HLSW can compress a maximum buffer size of 468kB in spectroscopy
SPU-HSR-OP0003	`dmch_cp.c, fill_out.c, datatx.c`	HLSW compresses DEC/MEC header lossless
SPU-HSR-OP0010	`Supervs.c`	DEC/MEC header field in spectroscopy (SPUID)
SPU-HSR-OP0020	`Supervs.c, csw.c, hk.c`	DEC/MEC header field in spectroscopy (Type)
SPU-HSR-OP0021	`Supervs.c`	DEC/MEC header field in spectroscopy (Type)
SPU-HSR-OP0030	`Supervs.c`	DEC/MEC header field in spectroscopy (CRDCR)

SPU-HSR-OP0031	`Supervs.c`	DEC/MEC header field in spectroscopy (CRDCR)
SPU-HSR-OP0040	`Supervs.c`	DEC/MEC header field in spectroscopy (RRR)
SPU-HSR-OP0041	`Supervs.c`	DEC/MEC header field in spectroscopy (RRR)
SPU-HSR-OP0050	`csw.c, hk.c, spec1_2.c`	DEC/MEC header field in spectroscopy (CMM)
SPU-HSR-OP0051	`supervs.c, csw.c, spec1_2.c, spec3.c, spec4.c, spec5.c, fill_in.c`	DEC/MEC header field in spectroscopy (CMM)
SPU-HSR-OP0060 deleted	`supervs.c, csw.c, spec1_2.c, spec3.c, spec4.c, spec5.c, fill_in.c`	DEC/MEC header field in spectroscopy (OBSID)
SPU-HSR-OP0070	`supervs.c`	DEC/MEC header field in spectroscopy (Validity)
SPU-HSR-OP0071 obsolete	`supervs.c, integ.c`	DEC/MEC header field in spectroscopy (Validity)
SPU-HSR-OP0072	`Supervs.c`	DEC/MEC header field in spectroscopy (Validity)
SPU-HSR-OP0080	`Supervs.c`	DEC/MEC header field in spectroscopy (CPR)
SPU-HSR-OP0081 obsolete	`Supervs.c, integ.c`	DEC/MEC header field in spectroscopy (CPR)
SPU-HSR-OP0082	`supervs.c, ramp_ft.c`	DEC/MEC header field in spectroscopy (CPR)
SPU-HSR-OP0083	`supervs.c`	DEC/MEC header field in spectroscopy (CPR)
SPU-HSR-OP0090	`supervs.c`	DEC/MEC header field in spectroscopy (Label)
SPU-HSR-OP0091	`supervs.c, csw.c, fill_in.c`	DEC/MEC header field in spectroscopy (Label)
SPU-HSR-OP0100	`supervs.c`	DEC/MEC header field in spectroscopy (GPR)
SPU-HSR-OP0101 obsolete	`supervs.c, integ.c`	DEC/MEC header field in spectroscopy (GPR)
SPU-HSR-OP0102	`supervs.c`	DEC/MEC header field in spectroscopy (GPR)
SPU-HSR-OP0110	`supervs.c, fill_out.c, datatx.c`	DEC/MEC header field in spectroscopy (BBID)
SPU-HSR-OP0120	`supervs.c, fill_out.c, datatx.c`	DEC/MEC header field in spectroscopy (Time)
SPU-HSR-OP0130	`supervs.c, fill_out.c, datatx.c`	DEC/MEC header field in spectroscopy (WPR)
SPU-HSR-OP0140	`supervs.c, fill_out.c, datatx.c`	DEC/MEC header field in spectroscopy (CRECR)
SPU-HSR-OP0200 obsolete	`supervs.c, csw.c, fill_in.c`	HLSW set the buffer granularity in photometry to 10 Hz in SWL and 2.5 Hz in LWL
SPU-HSR-OP0201	`supervs.c, csw.c, bol_ex.c, bol3.c, bol4.c, fill_in.c`	HLSW can compress a maximum buffer size of 480kB in photometry
SPU-HSR-OP0202	`dmch_cp.c, fill_out.c, datatx.c`	HLSW compresses DEC/MEC header lossless
SPU-HSR-OP0210	`supervs.c`	DEC/MEC header field in photometry (SPUID)
SPU-HSR-OP0220	`supervs.c, csw.c, hk.c`	DEC/MEC header field in photometry (Type)
SPU-HSR-OP0230	`supervs.c`	DEC/MEC header field in photometry (CRDCCP)
SPU-HSR-OP0231	`supervs.c`	DEC/MEC header field in photometry (CRDCCP)
SPU-HSR-OP0240	`supervs.c, fill_out.c, datatx.c`	DEC/MEC header field in photometry (CRC)
SPU-HSR-OP0250	`csw.c, hk.c, bol_ex.c`	DEC/MEC header field in photometry (CMM)
SPU-HSR-OP0251	`supervs.c, csw.c, bol_ex.c, bol3.c, bol4.c, fill_in.c`	DEC/MEC header field in photometry (CMM)
SPU-HSR-OP0260 deleted	`supervs.c, csw.c, bol_ex.c, bol3.c, bol4.c, fill_in.c`	DEC/MEC header field in photometry (OBSID)
SPU-HSR-OP0270	`supervs.c`	DEC/MEC header field in photometry (Validity)
SPU-HSR-OP0271	`average.c`	DEC/MEC header field in photometry (Validity)
SPU-HSR-OP0272	`average.c`	DEC/MEC header field in photometry (Validity)

SPU-HSR-OP0280	`integ.c`	DEC/MEC header field in photometry (CPR)
SPU-HSR-OP0281	`supervs.c`	DEC/MEC header field in photometry (CPR)
SPU-HSR-OP0290	`supervs.c`	DEC/MEC header field in photometry (Label)
SPU-HSR-OP0291	`supervs.c, csw.c, bol_ex.c, bol3.c, bol4.c, fill_in.c`	DEC/MEC header field in photometry (Label)
SPU-HSR-OP0300	`supervs.c`	DEC/MEC header field in photometry (DBID)
SPU-HSR-OP0310	`supervs.c, fill_out.c, datatx.c`	DEC/MEC header field in photometry (BBID)
SPU-HSR-OP0320	`supervs.c, fill_out.c, datatx.c`	DEC/MEC header field in photometry (Time)
SPU-HSR-OP0330	`supervs.c, fill_out.c, datatx.c`	DEC/MEC header field in photometry (WPR)
SPU-HSR-OP0340	`supervs.c, fill_out.c, datatx.c`	DEC/MEC header field in photometry (BSID)
SPU-HSR-OP0400	`supervs.c, csw.c, hk.c`	HLSW supports PACS operating modes
SPU-HSR-OP0401	`supervs.c`	Flexibility of the HLSW in science data compression
SPU-HSR-OP0402	`none` (see RD009)	mode parameters are not sent within DEC/MEC data
SPU-HSR-OP0500	`supervs.c, csw.c`	HLSW supports PACS observing modes
SPU-HSR-OP0501	`none` (see RD009)	mode parameters are not sent within DEC/MEC data
SPU-HSR-OP0600	`dxs.c, llc.c, fill_out.c, datatx.c, Rc_Sel.c`	Raw data from selected channels can be sent if required
SPU-HSR-OP0601	`dmch_cp.c, fill_out.c, datatx.c`	DEC/MEC header is compressed lossless apart in the buffer transmission mode
SPU-HSR-OP0602	`hk.c, fill_out.c, ramp_ft.c, average.c`	Specified parameters are included in the HLSW HK
SPU-HSR-OP0603	`supervs.c, spuasw.h, fill_out.c`	Nominal compressed entity time interval is 12s for LWL and 3s for SWL in photometry
SPU-HSR-OP0604	`supervs.c, spuasw.h, fill_out.c`	Nominal compressed entity time interval is 2s for LWL and SWL in spectroscopy
SPU-HSR-OP0610	`csw.c, bol_ex.c, spec1_2.c`	HLSW supports the PACS prime mode
SPU-HSR-OP0611	`hk.c, fill_out.c`	Compression results are included in the HLSW HK
SPU-HSR-OP0620	`csw.c, bol_ex.c, spec1_2.c`	HLSW supports the PACS parallel mode
SPU-HSR-OP0625	`csw.c, bol_ex.c`	HLSW supports the PACS prime mode
SPU-HSR-OP0630	`csw.c, bol3.c, spec3.c, dxs.c`	HLSW can compress data from selected detectors lossless
SPU-HSR-OP0631	`csw.c, bol3.c, spec3.c, dxs.c, T_S_Red.c`	HLSW can compress data from selected detectors lossless
SPU-HSR-OP0632	`csw.c, bol3.c, spec3.c, dxs.c, T_S_Red.c`	HLSW can compress data from selected detectors lossless
SPU-HSR-OP0640	`Spvs_Test.c, spu_tst.c, csw.c, phot_gn.c, spec_gn.c`	HLSW supports SPU test mode
SPU-HSR-OP0650	`csw.c, bol4.c, spec4.c`	HLSW can transmit data from selected detectors without compression
SPU-HSR-OP0660	`csw.c, buf_bol.c, buf_spec.c, fill_out.c`	HLSW supports buffer transmission mode
SPU-HSR-OP0661	`supervs.c, csw.c`	HLSW should be in stopped compression mode before running the buffer transmission mode
SPU-HSR-OP0670	`csw.c, spec5.c, spec1_2.c`	HLSW splits 4s ramp in two 2s ramps
SPU-HSR-RS0001	`All`	memory processing
SPU-HSR-RS0100	`All`	processing capacity
SPU-HSR-VF0001	`All`	SW acceptance verification

SPU-HSR-VF0002	`All` (+ decompression SW components)	ensure compression/decompression yields to orig. data
SPU-HSR-VF0003	`All`	ensure fully execution of algorithm logic by test data
SPU-HSR-AT0001	`All`	test cases for acceptance tests
SPU-HSR-DO0001	`All`	SDD replaces ADD, DDD and SRD
SPU-HSR-PT0001	`All`	Compiler G21K, Processor ADSP 21020, Communication Chip SMC S332, Operating System Virtuoso™
SPU-HSR-QA0001	`All`	HLSW development complies with the QA plan
SPU-HSR-QA0002	`All`	HLSW source code is commented
SPU-HSR-QA0100	`watchp.c`	handle anomalies with respect to command execution
SPU-HSR-QA0101	`All`	SW design to avoid flaws
SPU-HSR-QA0102	`All`	mechanisms to protect data and resources
SPU-HSR-QA0103	`spu_io.c, supervs.c, fill_in.c, hk.c`	handle RAM and EEPROM SEU
SPU-HSR-QA0104	`supervs.c`	handle corrupt and incomplete DEC/MEC data
SPU-HSR-QA0105	`asmlib.s`	detect and handle arithmetic errors
SPU-HSR-QA0106	`llc.c`	not lead to buffer overflow/underrun
SPU-HSR-QA0200	`All` (+ documentation)	maintainable to later extend
SPU-HSR-QA0201	`All` (+ documentation)	error detection and fault repair
SPU-HSR-QA0202	`make.bat` (in CVS: / obsw/ spu/ tools/ compilation)	deliverable scripts for SW build
SPU-HSR-QA0203	files in CVS: / obsw/ spu/ tools	SDE includes SW licences and tools
SPU-HSR-QA0204	HLSW.MAP (in CVS: / obsw/ spu/ tools/ compilation)	detailed map for SW deliveries
SPU-HSR-QA0205	files in CVS: / obsw/ spu/ tools	documentation of in-house tools
SPU-HSR-QA0206	see http://pacs.ster.kuleuven.ac.be/ shall and http://cvs.ster.kuleuven.be/	use of SPR/SCR and CVS management
SPU-HSR-SA0001	`All`	order of commands, HK and telemetry
SPU-HSR-SA0010	`hk.c`	HK provide detailed SPU status
SPU-HSR-SA0011	`hk.c, spu_io.c, fill_in.c`	Memory failure detection
SPU-HSR-SA0012	`hk.c`	DEC/MEC Header Errors

9 Software Requirements vs. Software Verification & Validation Traceability Matrix

Software Requirements	**Verification Activity** in RD017	**Description**
SPU-HSR-FN0001	Proc. 3.4.4.1 – 15	HLSW is able to receive command packets from DPU
SPU-HSR-FN0002	Proc. 3.4.4.1 – 15	HLSW is able to identify the DPU command
SPU-HSR-FN0003	Proc. 3.5.4.1 – 6 (spec) Proc. 3.6.4.1 – 6 (phot)	HLSW aborts data compression if DPU command is received
SPU-HSR-FN0004	Proc. 3.4.4.1	HLSW can load data into SPU RAM
SPU-HSR-FN0005	Proc. 3.4.4.1	HLSW checks the consistency of the data before the load
SPU-HSR-FN0006 transfered to **SPU-HSR-IF0011**	Proc. 3.4.4.1	Data load only to DRAM, Ext. DRAM and DPRAM
SPU-HSR-FN0007	Proc. 3.4.4.2	HLSW can dump data from SPU RAM
SPU-HSR-FN0008 transfered to **SPU-HSR-IF0021**	Proc. 3.4.4.2	Data dump only from PRAM, DRAM, Ext. DRAM and DPRAM
SPU-HSR-FN0009	Proc. 3.4.4.3	HLSW can check SPU RAM
SPU-HSR-FN0010	Proc. 3.4.4.3	Data check only for PRAM, DRAM, Ext. DRAM and DPRAM
SPU-HSR-FN0011	Proc. 3.4.4.4 – 15	HLSW checks consistency of the parameter of the perform activity command
SPU-HSR-FN0012	Proc. 3.4.4.4	HLSW can perform warm reset
SPU-HSR-FN0013	Proc. 3.5.4.7 – 16 (spec) Proc. 3.6.4.7 – 28 (phot)	HLSW can transmit data from selected channels in raw
SPU-HSR-FN0014	Proc. 3.4.4.6 – 8 (sim), Proc. 3.4.4.9 – 11 (sim) Proc. 3.5.4.1 – 4 & 6 (spec) Proc. 3.6.4.1 – 4 & 6 (phot)	HLSW can abort data compression
SPU-HSR-FN0015	Proc. 3.5.4.1 – 2 (spec) Proc. 3.6.4.1 – 2 (phot)	HLSW can start data compression
SPU-HSR-FN0016	Proc. 3.4.4.9 – 10	HLSW can generate simulated photometry data on board for test purposes
SPU-HSR-FN0017	Proc. 3.4.4.6 – 7	HLSW can generate simulated spectroscopy data on board for test purposes
SPU-HSR-FN0018 deleted	N/A	HLSW provides an interface to the peak-up program
SPU-HSR-FN0019 deleted	N/A	HLSW provides an interface to the bolometer background cancelling program
SPU-HSR-FN0020	Proc. 3.4.4.6, Proc. 3.4.4.9	HLSW checks consistency of the parameter of the write command
SPU-HSR-FN0021	Proc. 3.6.4.7 – 10	HLSW can load detectors table from subarray 1 in photometry
SPU-HSR-FN0022	Proc. 3.6.4.11 – 14	HLSW can load detectors table from subarray 2 in photometry

SPU-HSR-FN0023	Proc. 3.6.4.15 – 18	HLSW can load detectors table from subarray 3 in photometry
SPU-HSR-FN0024	Proc. 3.6.4.19 – 22	HLSW can load detectors table from subarray 4 in photometry
SPU-HSR-FN0025	Proc. 3.6.4.23 – 26	HLSW can load detectors table from subarray 5 in photometry
SPU-HSR-FN0026	Proc. 3.5.4.7 – 10	HLSW can load detectors table from subarray 1 in spectroscopy
SPU-HSR-FN0027	Proc. 3.6.4.11 – 14	HLSW can load detectors table from subarray 2 in spectroscopy
SPU-HSR-FN0028	Proc. 3.4.4.13	HLSW can load detector constants table in photometry
SPU-HSR-FN0029	Proc. 3.4.4.12	HLSW can load detector constants table in spectroscopy
SPU-HSR-FN0030	Proc. 3.4.4.6 (spec) Proc. 3.4.4.9 (phot)	HLSW can load simulated data table for SPU test mode
SPU-HSR-FN0031	Proc. 3.4.4.12	HLSW can copy SW code from RAM to EEPROM
SPU-HSR-FN0032	Proc. 3.2 (item 16)	HLSW connects to DEC/MEC upon DPU command
SPU-HSR-FN0100	Proc. 3.5.4.1 – 2 (spec) Proc. 3.6.4.1 – 2 (phot)	HLSW is able to receive data packets from DMC
SPU-HSR-FN0101	Proc. 3.5.4.1 – 6 (spec) Proc. 3.6.4.1 – 6 (phot)	HLSW can start data compression
SPU-HSR-FN0102	Proc. 3.5.4.1 – 2 (spec) Proc. 3.6.4.1 – 2 (phot)	HLSW can distinguish between photometry and spectroscopy data
SPU-HSR-FN0103	Proc. 3.5.4.18 – 20 (spec) Proc. 3.6.4.30 – 32 (phot)	HLSW is able to buffer DEC/MEC data to memory
SPU-HSR-FN0104	Proc. 3.5.4.1 – 6 (spec) Proc. 3.6.4.1 – 6 (phot)	For real time processing procedures are implemented in HLSW to handle the buffers
SPU-HSR-FN0105	Proc. 3.5.4.1 – 23 (spec) Proc. 3.6.4.1 – 35 (phot)	HLSW supports eleven compression modes
SPU-HSR-FN0200	Proc. 3.5.4.1 – 6 (spec) Proc. 3.6.4.1 – 6 (phot)	HLSW send permanently HK packets to DPU
SPU-HSR-FN0201	Proc. 3.5.4.1 – 6 (spec) => HK analysis Proc. 3.6.4.1 – 6 (phot) => HK analysis	„Are You Alive“ SPU is implemented in the HLSW HK
SPU-HSR-FN0202	Proc. 3.5.4.1 – 6 (spec) => HK analysis Proc. 3.6.4.1 – 6 (phot) => HK analysis	EDAC check status is implemented in the HLSW HK
SPU-HSR-FN0203	Proc. 3.5.4.1 – 6 (spec) => HK analysis Proc. 3.6.4.1 – 6 (phot) => HK analysis	Ground people are responsible for analyzing the failing memory addresses detected using EDAC check
SPU-HSR-FN0300	Proc. 3.5.4.1 – 6 (spec) Proc. 3.6.4.1 – 6 (phot)	HLSW fulfils real time data processing requirement
SPU-HSR-FN0301	Proc. 3.5.4.1 – 6 (spec) => TM Packet analysis Proc. 3.6.4.1 – 6 (phot) => TM Packet analysis	HLSW doesn’t allow science data loss
SPU-HSR-IF0001	Proc. 3.5.4.1 – 6 (spec) Proc. 3.6.4.1 – 6 (phot)	HLSW aborts any running activity, if DPU command is received
SPU-HSR-IF0002	Proc. 3.4.4.1 – 15	Maximum response time for DPU commands is 200ms
SPU-HSR-IF0003	Proc. 3.4.4.1 – 15	HLSW acknowledges any received DPU command according to the defined protocol
SPU-HSR-IF0010	Proc. 3.4.4.1	HLSW acknowledges the execution of the load command
SPU-HSR-IF0011	Proc. 3.4.4.1	Data load only to DRAM, Ext. DRAM and DPRAM

SPU-HSR-IF0020	Proc. 3.4.4.2	HLSW acknowledges the execution of the dump command
SPU-HSR-IF0021	Proc. 3.4.4.2	Data dump only from PRAM, DRAM, Ext. DRAM and DPRAM
SPU-HSR-IF0030	Proc. 3.4.4.3	HLSW acknowledges the execution of the check command
SPU-HSR-IF0040 deleted	N/A	HLSW acknowledges the reception of the peak-up command
SPU-HSR-IF0050	Proc. 3.4.4.4	HLSW acknowledges the reception of the warm reset command
SPU-HSR-IF0060	Proc. 3.4.4.5	HLSW acknowledges the execution of the raw channel selection command
SPU-HSR-IF0070	Proc. 3.4.4.8, Proc. 3.4.4.11	HLSW acknowledges the reception of the stop command
SPU-HSR-IF0080	Proc. 3.5.4.2 (spec) Proc. 3.6.4.2 (phot)	HLSW acknowledges the reception of the start command
SPU-HSR-IF0090	Proc. 3.4.4.10	HLSW acknowledges the reception of the SPU test in photometry command
SPU-HSR-IF0100	Proc. 3.4.4.7	HLSW acknowledges the reception of the SPU test in spectroscopy command
SPU-HSR-IF0110 deleted	N/A	HLSW acknowledges the reception of the Bolometer Background Cancelling command
SPU-HSR-IF0120	Proc. 3.4.4.6, Proc. 3.4.4.9	HLSW acknowledges the execution of the write command
SPU-HSR-IF0130	Proc. 3.4.4.12	HLSW acknowledges the reception of the Copy Data to EEPROM command
SPU-HSR-IF0131	Proc. 3.4.4.12	No HK packet are transmitted to DPU while data are copied to EEPROM
SPU-HSR-IF0140	Proc. 3.2 (item 16)	HLSW acknowledges the reception of the Connect to DEC/MEC command
SPU-HSR-IF0200	Proc. 3.4.4.6 / 7 (sim), Proc. 3.4.4.9 / 10 (sim) Proc. 3.5.4.1 – 2 (spec) Proc. 3.6.4.1 – 2 (phot)	HLSW is able to transmit TM packets to DPU
SPU-HSR-IF0201	Proc. 3.5.4.1 – 6 (spec) => TM Packet analysis Proc. 3.6.4.1 – 6 (phot) => TM Packet analysis	HLSW can buffer a maximum number of 75 TM packets
SPU-HSR-IF0300	Proc. 3.5.4.1 – 6 (spec) Proc. 3.6.4.1 – 6 (phot)	HLSW doesn't acknowledge the reception of DEC/MEC packets
SPU-HSR-IF0301	Proc. 3.5.4.1 – 6 (spec) => HK analysis Proc. 3.6.4.1 – 6 (phot) => HK analysis	HLSW can detect some corrupted fields in the DEC/MEC packets
SPU-HSR-IF0302	Proc. 3.5.4.1 – 6 (spec) Proc. 3.6.4.1 – 6 (phot)	HLSW can distinguish between photometry and spectroscopy packets
SPU-HSR-IF0303	Proc. 3.5.4.1 – 6 (spec) => HK analysis Proc. 3.6.4.1 – 6 (phot) => HK analysis	DEC/MEC link connection status is set in the HLSW HK
SPU-HSR-IF0304	Proc. 3.5.4.1 – 6 (spec) => HK analysis Proc. 3.6.4.1 – 6 (phot) => HK analysis	DEC/MEC packet size
SPU-HSR-OP0001	Proc. 3.5.4.1 – 6 => TM analysis	HLSW set the buffer granularity in spectroscopy to 8 Hz
SPU-HSR-OP0002	Proc. 3.5.4.1 – 6 => TM analysis	HLSW can compress a maximum buffer size of 468kB in spectroscopy

SPU-HSR-OP0003	Proc. 3.5.4.1 – 6 => TM analysis	HLSW compresses DEC/MEC header lossless
SPU-HSR-OP0010	*is not taken into account by HLSW*	DEC/MEC header field in spectroscopy (SPUID)
SPU-HSR-OP0020	Verifiable by code inspection	DEC/MEC header field in spectroscopy (Type)
SPU-HSR-OP0021	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (Type)
SPU-HSR-OP0030	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (CRDCR)
SPU-HSR-OP0031	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (CRDCR)
SPU-HSR-OP0040	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (RRR)
SPU-HSR-OP0041	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (RRR)
SPU-HSR-OP0050	Proc. 3.5.4.1 – 65	DEC/MEC header field in spectroscopy (CMM)
SPU-HSR-OP0051	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (CMM)
SPU-HSR-OP0060 deleted	Proc. 3.5.4.1 – 6	DEC/MEC header field in spectroscopy (OBSID)
SPU-HSR-OP0070	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (Validity)
SPU-HSR-OP0071	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (Validity)
SPU-HSR-OP0072	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (Validity)
SPU-HSR-OP0080	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (CPR)
SPU-HSR-OP0081	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (CPR)
SPU-HSR-OP0082	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (CPR)
SPU-HSR-OP0083	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (CPR)
SPU-HSR-OP0090	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (Label)
SPU-HSR-OP0091	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (Label)
SPU-HSR-OP0100	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (GPR)
SPU-HSR-OP0101	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (GPR)
SPU-HSR-OP0102	Proc. 3.5.4.1 – 6 => TM analysis	DEC/MEC header field in spectroscopy (GPR)
SPU-HSR-OP0110	*is not taken into account by HLSW*	DEC/MEC header field in spectroscopy (BBID)
SPU-HSR-OP0120	*is not taken into account by HLSW*	DEC/MEC header field in spectroscopy (Time)
SPU-HSR-OP0130	*is not taken into account by HLSW*	DEC/MEC header field in spectroscopy (WPR)
SPU-HSR-OP0140	*is not taken into account by HLSW*	DEC/MEC header field in spectroscopy (CRECR)
SPU-HSR-OP0200	Proc. 3.6.4.1 – 6 => TM analysis	HLSW set the buffer granularity in photometry to 10 Hz in SWL and 2.5 Hz in LWL
SPU-HSR-OP0201	Proc. 3.6.4.1 – 6 => TM analysis	HLSW can compress a maximum buffer size of 480kB in photometry
SPU-HSR-OP0202	Proc. 3.6.4.1 – 6 => TM analysis	HLSW compresses DEC/MEC header lossless
SPU-HSR-OP0210	*is not taken into account by HLSW*	DEC/MEC header field in photometry (SPUID)
SPU-HSR-OP0220	Proc. 3.6.4.1 – 6 => TM analysis	DEC/MEC header field in photometry (Type)
SPU-HSR-OP0230	Proc. 3.6.4.1 – 6 => TM analysis	DEC/MEC header field in photometry (CRDCCP)
SPU-HSR-OP0231	Proc. 3.6.4.1 – 6 => TM analysis	DEC/MEC header field in photometry (CRDCCP)
SPU-HSR-OP0240	*is not taken into account by HLSW*	DEC/MEC header field in photometry (CRC)
SPU-HSR-OP0250	Proc. 3.6.4.1 – 6	DEC/MEC header field in photometry (CMM)
SPU-HSR-OP0251	Proc. 3.6.4.1 – 6 => TM analysis	DEC/MEC header field in photometry (CMM)
SPU-HSR-OP0260 deleted	Proc. 3.6.4.1 – 6	DEC/MEC header field in photometry (OBSID)
SPU-HSR-OP0270	Proc. 3.6.4.1 – 6 => TM analysis	DEC/MEC header field in photometry (Validity)
SPU-HSR-OP0271	Proc. 3.6.4.1 – 6 => TM analysis	DEC/MEC header field in photometry (Validity)
SPU-HSR-OP0272	Proc. 3.6.4.1 – 6 => TM analysis	DEC/MEC header field in photometry (Validity)
SPU-HSR-OP0280	Proc. 3.6.4.1 – 6 => TM analysis	DEC/MEC header field in photometry (CPR)
SPU-HSR-OP0281	Proc. 3.6.4.1 – 6 => TM analysis	DEC/MEC header field in photometry (CPR)

SPU-HSR-OP0290	Proc. 3.6.4.1 – 5 => TM analysis	DEC/MEC header field in photometry (Label)
SPU-HSR-OP0291	Proc. 3.6.4.1 – 5 => TM analysis	DEC/MEC header field in photometry (Label)
SPU-HSR-OP0300	*is not taken into account by HLSW*	DEC/MEC header field in photometry (DBID)
SPU-HSR-OP0310	*is not taken into account by HLSW*	DEC/MEC header field in photometry (BBID)
SPU-HSR-OP0320	*is not taken into account by HLSW*	DEC/MEC header field in photometry (Time)
SPU-HSR-OP0330	*is not taken into account by HLSW*	DEC/MEC header field in photometry (WPR)
SPU-HSR-OP0340	*is not taken into account by HLSW*	DEC/MEC header field in photometry (BSID)
SPU-HSR-OP0400	3.5.4.1 – 32 (spec) 3.6.4.1 – 41 (phot)	HLSW supports PACS operating modes
SPU-HSR-OP0401	3.5.4.1 – 32 (spec) 3.6.4.1 – 41 (phot)	Flexibility of the HLSW in science data compression
SPU-HSR-OP0402	*is not taken into account by HLSW*	mode parameters are not sent within DEC/MEC data
SPU-HSR-OP0500	3.5.4.1 – 32 (spec) 3.6.4.1 – 41 (phot)	HLSW supports PACS observing modes
SPU-HSR-OP0501	*is not taken into account by HLSW*	mode parameters are not sent within DEC/MEC data
SPU-HSR-OP0600	Proc. 3.5.4.1 – 2 (spec) Proc. 3.6.4.1 – 2 (phot)	Raw data from selected channels can be sent if required
SPU-HSR-OP0601	Proc. 3.5.4.1 – 32 (spec) => TM analysis Proc. 3.6.4.1 – 41 (phot) => TM analysis	DEC/MEC header is compressed lossless apart in the buffer transmission mode
SPU-HSR-OP0602	Proc. 3.5.4.1 – 32 (spec) => HK analysis Proc. 3.6.4.1 – 41 (phot) => HK analysis	Specified parameters are included in the HLSW HK
SPU-HSR-OP0603	Proc. 3.6.4.1 – 32 => TM analysis	Nominal compressed entity time interval is 12s for LWL and 3s for SWL in photometry
SPU-HSR-OP0604	Proc. 3.5.4.1 – 41 => TM analysis	Nominal compressed entity time interval is 2s for LWL and SWL in spectroscopy
SPU-HSR-OP0610	Proc. 3.5.4.1 – 2 (spec) Proc. 3.6.4.1 – 2 (phot)	HLSW supports the PACS prime mode
SPU-HSR-OP0611	Proc. 3.5.4.1 – 32 (spec) => HK analysis Proc. 3.6.4.1 – 41 (phot) => HK analysis	Compression results are included in the HLSW HK
SPU-HSR-OP0620	Proc. 3.5.4.1 – 2 (spec) Proc. 3.6.4.1 – 2 (phot)	HLSW supports the PACS parallel mode
SPU-HSR-OP0625	Proc. 3.5.4.1 – 32 (spec) Proc. 3.6.4.1 – 41 (phot)	HLSW supports the PACS prime mode
SPU-HSR-OP0630	Proc. 3.5.4.7 – 10 (spec) Proc. 3.6.4.7 – 10 (phot)	HLSW can compress data from selected detectors lossless
SPU-HSR-OP0631	Proc. 3.5.4.7 – 16 (spec) Proc. 3.6.4.7 – 28 (phot)	HLSW can compress data from selected detectors lossless
SPU-HSR-OP0632	Proc. 3.5.4.7 – 10 (spec) Proc. 3.6.4.7 – 10 (phot)	HLSW can compress data from selected detectors lossless
SPU-HSR-OP0640	Proc. 3.4.4.6 – 7 (sim spec) Proc. 3.4.4.9 – 10 (sim phot)	HLSW supports SPU test mode
SPU-HSR-OP0650	Proc. 3.5.4.11 – 17 (spec) Proc. 3.6.4.23 – 29 (phot)	HLSW can transmit data from selected detectors without compression
SPU-HSR-OP0660	Proc. 3.5.4.11 – 20 (spec) Proc. 3.6.4.23 – 32 (phot)	HLSW supports buffer transmission mode
SPU-HSR-OP0661	Proc. 3.5.4.11 – 20 (spec) Proc. 3.6.4.23 – 32 (phot)	HLSW should be in stopped compression mode before running the buffer transmission mode

SPU-HSR-OP0670	Proc. 3.5.4.21 – 23 (spec)	HLSW splits 4s ramp in two 2s ramps
SPU-HSR-RS0001	Proc. 3.5.4.1 – 32 (spec) => TM/HK analysis Proc. 3.6.4.1 – 41 (phot) => TM/HK analysis	20% more memory capacity
SPU-HSR-RS0100	Proc. 3.5.4.1 – 32 (spec) => TM/HK analysis Proc. 3.6.4.1 – 41 (phot) => TM/HK analysis	20% spare processing capacity
SPU-HSR-VF0001	All	SW acceptance verification
SPU-HSR-VF0002	All (+ PCSS test harness)	ensure compression/decompression yields to orig. data
SPU-HSR-VF0003	All	ensure fully execution of algorithm logic by test data
SPU-HSR-AT0001	All	test cases for acceptance tests
SPU-HSR-DO0001	see RD001	SSD replaces ADD, DDD and SRD
SPU-HSR-PT0001	see RD016	Compiler G21K, Processor ADSP 21020, Communication Chip SMC S332, Operating System Virtuoso^{TM}
SPU-HSR-QA0001	Verifiable by code inspection	HLSW development complies with the QA plan
SPU-HSR-QA0002	Verifiable by code inspection	HLSW source code is commented
SPU-HSR-QA0100	Proc. 3.5.4.1 – 6 (spec) Proc. 3.6.4.1 – 6 (phot)	handle anomalies with respect to command execution
SPU-HSR-QA0101	All	SW design to avoid flaws
SPU-HSR-QA0102	All	mechanisms to protect data and resources
SPU-HSR-QA0103	Verifiable by code inspection	handle RAM and EEPROM SEU
SPU-HSR-QA0104	Proc. 3.5.4.1 – 32 (spec) => TM/HK analysis Proc. 3.6.4.1 – 41 (phot) => TM/HK analysis	handle corrupt and incomplete DEC/MEC data
SPU-HSR-QA0105	Verifiable by code inspection	detect and handle arithmetic errors
SPU-HSR-QA0106	Verifiable by code inspection	not lead to buffer overflow/underrun
SPU-HSR-QA0200	Verifiable by code inspection (+ see RD016)	maintainable to later extend
SPU-HSR-QA0201	Verifiable by code inspection (+ see RD016)	error detection and fault repair
SPU-HSR-QA0202	N/A	deliverable scripts for SW build
SPU-HSR-QA0203	N/A	SDE includes SW licences and tools
SPU-HSR-QA0204	N/A	detailed map for SW deliveries
SPU-HSR-QA0205	N/A	documentation of in-house tools
SPU-HSR-QA0206	N/A	use of SPR/SCR and CVS management
SPU-HSR-SA0001	Verifiable by code inspection (+ see RD016)	order of commands, HK and telemetry
SPU-HSR-SA0010	Proc. 3.5.4.1 – 32 (spec) => TM/HK analysis Proc. 3.6.4.1 – 41 (phot) => TM/HK analysis	HK provide detailed SPU status
SPU-HSR-SA0011	Proc. 3.5.4.1 – 32 (spec) => TM/HK analysis Proc. 3.6.4.1 – 41 (phot) => TM/HK analysis	Memory failure detection
SPU-HSR-SA0012	Proc. 3.5.4.1 – 32 (spec) => TM/HK analysis Proc. 3.6.4.1 – 41 (phot) => TM/HK analysis	DEC/MEC Header Errors

Note: N/A indicates that this requirements could not be tested or visualised at SVV level (HK, TM or code insepction) or belongs to SW management or deleted SW tasks.

Appendices

Appendix A: LLC Algorithms

A.1 Currently implemented Algorithms

A.1.1 Lossless DEC/MEC Header Compression

The actually implemented algorithms in SPU HLSW version 13.8 are:

1.) D_S_TRR

2.) ZRS

3.) RZIP

4.) RZIP2

See Appendix A.2 for more detail. These algorithms have been tested for the worst case with several variations in the DEC/MEC header. The maximum compressed header size will not exceed 0.5 Kbytes.

A.1.2 Lossless Science Data Compression

The actually implemented lossless compression algorithms in SPU HLSW version 13.8 are:

1.) Photometry

a. Temporal and Spatial redundancy Reduction

b. VBWL encoder

c. FM Arithmetic Coding

2.) Specstropcopy

a. Data Sorting and Spatial Redundancy Reduction

b. pacs_codec

c. RZIP (with range 3)

d. Temporal Redundancy Reduction

e. Rampdiff

f. FM Arithmetic Coding

See Appendix A.2 for more details.

A.1.3 Raw Channel Data Compression

The actually implemented Raw Channel compression algorithms in SPU HLSW version 13.8 are:

1.) Photometry

a. D_S_TRR

b. RZIP (with range 7)

c. RZIP (with range 6)

d. RZIP (with range 4)

2.) Specstropcopy

a. Data Sorting and Spatial Redundancy Reduction

b. RZIP (with range 3)

See Appendix A.2 for more details.

A.2. Library of LLC Algorithms

A.2.1 List of Available Algorithms

The following table shows a summary of possible LLC algorithms.

Name of Algorithm	Description	Implemented in HLSW v13.8
Dynamic and Static Temporal Redundancy Reduction (D_S_TRR)	see section A.2.2	☑
Simple Zero Repetition Suppression (ZRS)	see section A.2.3	☑
RZIP/RZIP2	see section A.2.4	☑
Arithmetic Coding (ARC)	see section A.2.5	☑
Data Resorting (Reorder)	see section A.2.6	☑
PACS Codec (new_srt)	see section A.2.7	☑
FM Aritmetic Compression	see section A.2.8	☑
VBWL_short	see section A.2.9	☑

Table 13. List of available LLC algorithms.

A.2.2 Dynamic and Static Temporal Redundancy Reduction (D_S_TRR)

We consider 512 frames whereas each contains 13 words of data (DMCH). Then the algorithm of the DTRR is as follows:

```
Frame001 is not changed
Frame002new = Frame001 – Frame002
Frame003new = Frame002 – Frame003
Frame004new = Frame003 – Frame004
…
Frame512new = Frame511 – Frame512
```

In the next step the STRR algorithm is done:

```
Frame001 is not changed
Frame002new = Frame001 – Frame002
Frame003newnew = Frame002new – Frame003new
Frame004newnew = Frame002new – Frame004new
…
```

```
Frame512newnew = Frame002new – Frame512new
```

A.2.3 Simple Zero Repetition Suppression (ZRS)

ZRS is a small algorithm to quickly get rid of buffers with many zeroes. It simply ignores zeroes by encoding non-zero symbols and their relative position in the original buffer - or, in other words, the number of zeroes between the symbol and the one that was encoded before.

A symbol sequence of

```
A 0 B 0 0 0 0 0 0 0 C 0 0 0 A
```

Is encoded to

```
A 0 B 1 C 7 A 3
```

Note that the first symbol's position information is always 0. Another thing is that a buffer, which does not at least contain as many 0s as other symbols, is not compressed but expanded.

A.2.4 RZIP/RZIP2

Rzip is a simple lexicographic compression algorithm written for PACS OBSW DMC header compression (see RD018). The goal is to achieve maximum compression with a minimum of processing power on a 32-Bit big endian DSP. Though its power lies within iteration, it is considerably faster than conventional lz-based algorithms and much more efficient than encoders using DPCM and VBWL.

Let `SOURCE` be a buffer of data to compress with a size of `SSIZE`.
Let `DEST` be the destination buffer where the compressed data will be put.
Let `ALPHA` be a working buffer of `SSIZE` (it could be `SSIZE` bits, but this would be slow).

`SYMBOLSIZE` is a parameter, which determines the number of bits to use for the symbols.
`RANGEWIDTH` is a parameter, which determines the number of bits to use for encoding ranges.
A width of 3 means that the `RANGE` will be 7 (2^3-1).

Let the `SYMBOLSIZE` be 32bit. Let our `SSIZE` be 128 Bytes. Hence the `SOURCE` consists of 128*8/32 symbols. Add the necessary empty bits to the buffer if it is not separable by the `SYMBOLSIZE`. Store the number of bits added that way in `TAIL`, which will contribute to the compressed header information. In our example we have 32 symbols a 32 bit and the tail is 0.

We will now take a look at the symbols. Our buffer may look like:

`A-A-B-C-A-B-A-C-D-… (32 32bit Symbols)`

`0-0-0-0-0-0-0-0-0-… (This is our alpha channel)`

a) Select the first unused (alpha=0) Symbol. This is shown **bold** above. Set the Alpha 1.

b) Look ahead if you find the symbol in the next symbols within `RANGE`. If you do so, code `YES` in 1 bit plus the `RANGE` within `RANGEWIDTH` bits. Set the Alpha of the found symbol 1. Reset the `RANGE` to 0 and continue until no further occurrences are found. Code `NO` in 1 bit.

c) Go back to a) until the buffer is finished.

d) Encode the original size plus the tail in the header. You may also encode a version number, the symbol size and the range width optionally.

NOTE THAT:

The difference between `A-A` will be encoded 0 (0 symbols are between them).

The difference between `A-X-X-A` will be encoded 2.

Once you have already coded all As, the difference between the Bs in

`B-A-C-B` will be encoded as 1 (the As are already invisible due to the ALPHA).

Our DEST looks like:

```
A-Y0-Y2-Y1-…-N
B-Y1-…-N
C-Y0-…-N
D…
```

Iteration:

Once you have coded your buffer with a set of parameters, it can be encoded another time with different parameters. For example, for DMC compression a 32-5 32-3 32-5 triple run showed the best compression.

The following keys are used for compression-decompression interface to identify the different RZIP combination.

RZIP Combination (in term of range)	Key
3	0x30xxxxxx
3-3	0x22xxxxxx
3-3-3	0x03xxxxxx
7	0x70xxxxxx
6-6	0x00xxxxxx
7-6-4	0x07xxxxxx

Table 14. Keys for RZIP Combinationss

RZIP2 works in principle in the same way as RZIP, except that the symbols, answers and ranges are stored in a separated way. First, the 32-bit symbols are stored, then the block of ranges, padded by the block of answers. That way better iterative characteristics are achieved.

A.2.5 Arithmetic Coding (ARC)

The algorithm for encoding data using arithmetic coding works conceptually as follows:

- The encoding process begins with a current interval [L ; H[initialised to [0 ; 1[.
- For each symbol of the file, two steps are performed:
  - o The current interval is divided into subintervals, one for each possible alphabet symbol. The size of a symbol's subinterval is proportional to the estimated probability that the symbol will be the next symbol in the file, according to the model of the input.
  - o The subinterval corresponding to the symbol that actually occurs next in the file is selected as the new current interval.
- Enough bits are output to distinguish the final current interval from all other possible final intervals.

The length of the final subinterval is clearly equal to the product of the probabilities of the individual symbols, which is the probability p of the particular sequence of symbols in the file. The final step uses almost exactly - $\log_2$ p bits to distinguish the file from all other possible files. An additional mechanism is needed to indicate the end of the file, either a special end-of-file symbol coded just once, or some external indication of the file's length.

In the second step, it is required to compute only the subinterval corresponding to the symbol that actually occurs. To do this, two cumulative probabilities are defined, $Pc_i = \Sigma p_k$ (for k = 1 to i-1) (the cumulative probability) and $Pn_i = \Sigma p_k$ (for k = 1 to i) (the next cumulative probability). The new subinterval is [$L+Pc_i$ (H-L), $L+Pn_i$ (H-L)]. The need to maintain and supply cumulative probabilities requires the model to have a complicated data structure.

Example: a non-adaptive code, encoding the stream bbb using arbitrary fixed probability estimates $p_a = 0.4$, $p_b = 0.5$, and $p_{EOF} = 0.1$.

Encoding proceeds as shown in Table 15.

Current Interval	Action	Subinterval a	Subinterval b	Subinterval EOF	Input
[0.0000 ;1.0000[	Subdivide	[0.0000 ; 0.4000[	[0.4000 ; 0.9000[	[0.9000 ; 1.0000[	b
[0.4000 ;0.9000[	Subdivide	[0.4000 ; 0.6000[	[0.6000 ; 0.8500[	[0.8500 ; 0.9000[	b
[0.6000 ;0.8500[	Subdivide	[0.6000 ; 0.7000[	[0.7000 ; 0.8250[	[0.8250 ; 0.8500[	b
[0.7000 ;0.8250[	Subdivide	[0.7000 ; 0.7500[	[0.7500 ; 0.8125[	[0.8125 ; 0.8250[	EOF
[0.8125 ;0.8250[	Subdivide				

Table 15. The Basic Arithmetic Coding Process

The final interval is [0.8125, 0.8250[, which in binary is approximately [0.11010 00000, 0.11010 01100[. This interval is uniquely identified by outputting 11010 00. According to the fixed model, the probability p of this particular file is $(0.5)^3$ (0.1) = 0.0125 (exactly the size of the final interval) and the code length (in bits) should be -log2 p ≈ 6.322. In practice, 7 bits are needed.

For further details see AD005.

A.2.6 Data Resorting (Reorder)

The aim of this algorithm is to increase the local redundancy (between successive samples) of the data but resorting the samples.

We consider 64 ramps from 450 detectors wher each ramp contains 8 samples. The Reorder algorithm sort the data as follows:

1st sample of 1st ramp of 1st detector

1st sample of 2nd ramp of 1st detector

…

1st sample of 64th ramp of 1st detector

1st sample of 1st ramp of 2nd detector

1st sample of 2nd ramp of 2nd detector

…

1st sample of 64th ramp of 2nd detector

….

1st sample of 64th ramp of 450th detector

2nd sample of 1st ramp of 1st detector

2nd sample of 2nd ramp of 1st detector

…

2nd sample of 64th ramp of 1st detector

……

8th sample of 1st ramp of 450th detector

8th sample of 2nd ramp of 450th detector

…

8th sample of 64th ramp of 450th detector

A.2.7 PACS Codec (new_srt)

This algorithm is based on szip, which uses a Burrows-Wheeler Transform like block sorting technique, followed by a dynamic model range coder.

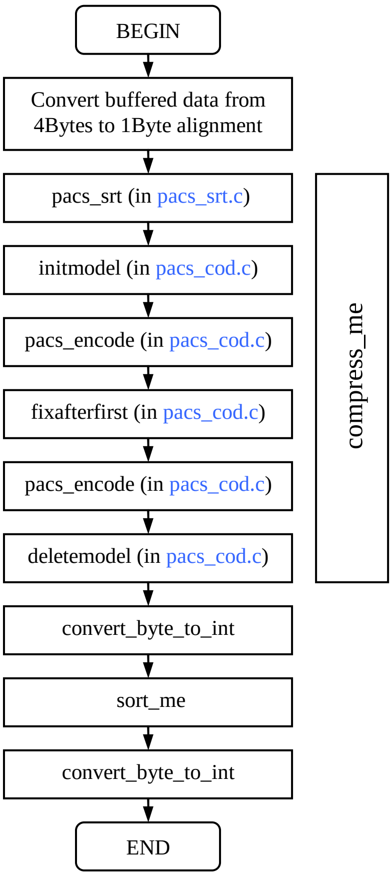


Figure 69. Scheme of the PACS Codec algorithm.

A.2.8 FM Arithmetic Compression

The compression scheme has been adapted to address the data from the FM detectors in a better way. In principle, the encoding process is still the same as in the standard Arithmetic Compression, with a few modifications:

- the chunksize is fixed at 8192 byte to replace the division by a right shift.
- The input symbol width is 32 bit

- Input values bigger than 255 are encoded with the “spill symbol” and sent to the “spillover”

- The “spillover” is a separate buffer which will be compressed by VBWL. The VBWL algorithm used is a slight modification of in the VBWL only setting.

- The data model is semi-static: The fitst chunk is compressed with a built-in statistic, each next chunk is compressed with a statistic derived from the previous chunk.

A.2.9 VBWL

A standard VBWL implementation with a few modifications, allowing for better flexibility. The blocksize is pretty short (4 symbols). Three different code lengths are used: the standard length is 3 bits, addressing the majority of the encoded (positive) differences. A 7-bit length is used for encoding bigger differences.

Appendix B: Fitting Algorithms

B.1 Currently implemented Algorithms

B.1.1 Robust Averaging in Photometry

The actually implemented algorithm in SPU HLSW version 12.2 is:

1.) Mean Value Calculation for signed or unsigned integers (see Appendix B.2.1.2).

B.1.2 Ramp-Fitting in Spectroscopy

The actually implemented algorithm in SPU HLSW version 12.2 is:

1.) Least Squares Fit for more than 2 samples to fit (default) (see Appendix B.2.2.2)

2.) Two-sample difference (see Appendix B.2.2.7) if least square fit if selected and 2 samples per subramps are considered

3.) Mean algorithm if selected (see Appendix B.2.1.2)

B.2 Library of Fitting Algorithms

B.2.1 List of Available Algorithms

The following table shows a summary of possible fitting algorithms for robust averaging in photometry and ramp-fitting in spectroscopy.

Action	Name of Algorithm	Description	Implemented in HLSW v13.8
Photometry			
Glitch Detection and Rejection	Sample Deviation Detection Method	see section B.2.2.1.1	☑
	Averaging with Median + Mean Value Calculation	see section B.2.2.1.2	☐
Averaging	Mean Value Calculation	see section B.2.2.2.1	☑
Spectroscopy			
Glitch Detection and Rejection	Crossed Slope Deviation Detection	see section B.2.3.1.1	☑
Ramp-Fitting	Least Squares Fitting	see section B.2.3.2.1	☑
	Two-Sample-Fit	see section B.2.3.2.4	☑
	Slope Deviation Detection Method	see section B.2.3.2.5	☐
	Two-Samples Difference Method	see section B.2.3.2.6	☑
	Subramp Fitting	see section B.2.3.2.7	☑
Averaging	Mean Value Calculation	see section B.2.2.2.1	☑

Table 16. List of available fitting algorithms.

B.2.2 Robust Averaging in Photometry

B.2.2.1 Glitch Detection and Rejection

B.2.2.1.1 Sample Deviation Detection Method

The glitch detection uses the Sample Deviation Detection Method. It performs the following:

1- Calculate the sum of valid quadruples.

S(i DIV 4) = sum(X(4*(i DIV 4):4*(i DIV 4)+3))

2- Calculate the deviation (using the weighted sample) of every sample in the quadruple comparing to the calculated sum.

Dev(i) = 4X(i) – S(i DIV 4)

If no glitches occur, all deviations would be less than a predefined Threshold

3- Remove all deviations that are above a certain threshold

4- Average all the remaining readouts with the currently implemented averaging algorithm.

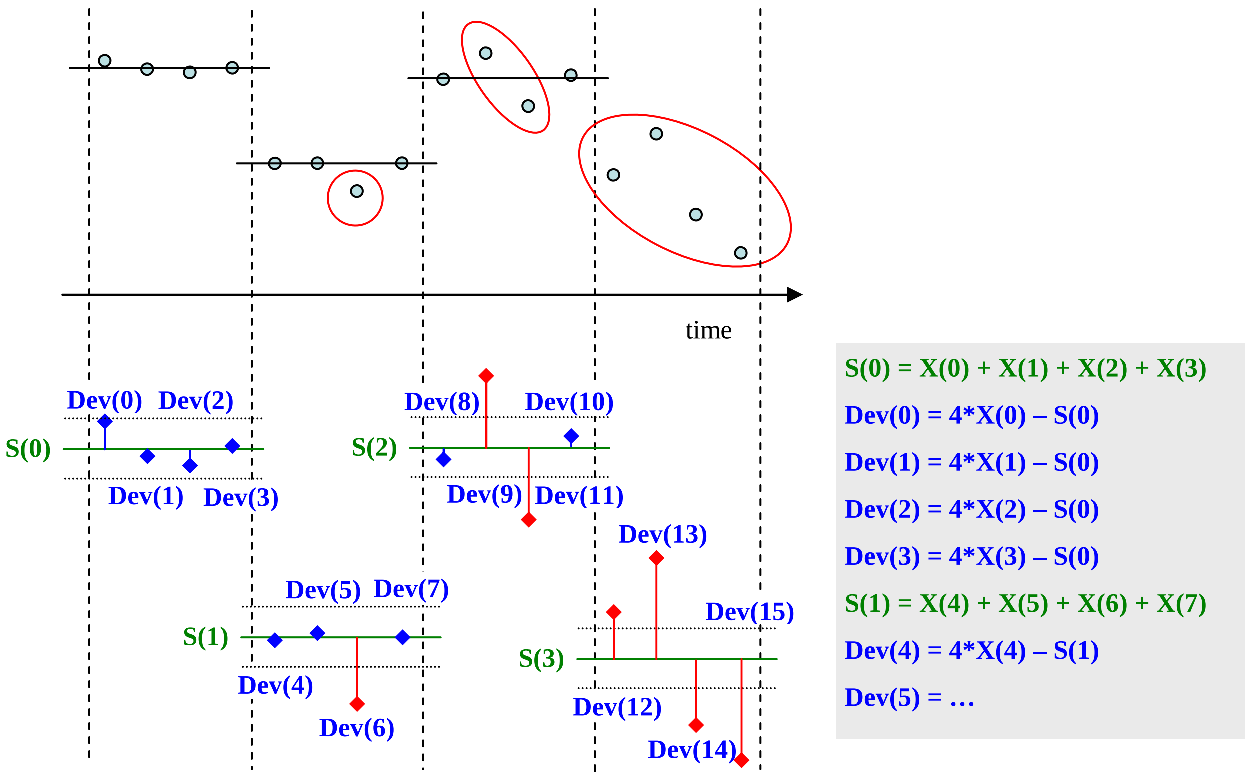


Figure 70. Example showing the sample deviation detection method

B.2.2.2 Averaging

B.2.2.2.1 FM Averaging algorithm

Beginning with software version 13.9, photometry and spectroscopy both use the same statistical averaging algorithm for the calculation of mean values. It supports arbitrary number of samples to average and additional bit rounding at the same time. There is also a random number generator involved for statistically correct up/down-rounding.

The mean value of this averaging method is generated as follows:

$$mean = \frac{1}{2^r \cdot n}\sum_{i=1}^{n} y_i + \frac{1}{2}\operatorname{sgn}(\sum_{i=1}^{n} y_i) + \frac{rand(1)}{2^r \cdot n}$$

Where n is the number of samples to average, r is the number of bits to round, sgn() is the signum function and rand(1) is a random number being 0 or 1. After calculation, the result is cast to an integer. Note that the mean is right shifted yb r bits if bit rounding is used.

For both, spectroscopy and photometry it is possible to use the "old" averaging algorithms via a parameter in the detector constants photometry/spectroscopy table.

B.2.2.2.2 Mean Value Calculation

This averaging method consists of calculating of the mean value. All points in the support set are taken for the mean calculation. The advantage of this method is the simplicity of the mean value calculation. One assumption on the support set is that any kind of drift, skewness, detector behaviour, etc. is corrected by the preprocessing step.

The main objective of the averaging method is a robust oversampling reduction to facilitate the data processing. Further data processing can be performed on ground.

The mean value of this averaging method is generated for 4 data values as follows

$$mean = \frac{(y(x) + y(x+1) + y(x+2) + y(x+3) + 2)}{4}$$

where y represent either signed or unsigned integer values. In the software the mean is calculated like below.

Pseudo-Code example:

```
                    Mean_val= ((y[x] + y[x+1] + y[x+2] + y[x+3]) + 2) >> 2

Definitions :
signed :        int Mean_val, y;
                unsigned int x;

unsigned :      unsigned int Mean_val, y, x ;
```

After the summation a bitwise shift by two was performed. This is equal to a division by 4. The result of this mean value calculation is either signed or unsigned integer. The remainder of the division is truncated.

B.2.3 Ramp-Fitting in Spectroscopy

B.2.3.1 Glitch Detection and Rejection

B.2.3.1.1 Crossed Slope Deviation Detection

The glitch detection and rejection using the Wide Slope Deviation Detection method (*Figure 71*) is performed as described below:

1- Calculate the differences between samples in a ramp as follows:

D(i,dist) = X(i) – X(i+dist)

2- Calculate the deviation of these differences as follows.

Dev(i,dist) = D(i,dist) – dist * D(i,1)

For the first sub-ramp the difference of the first two samples is compared with the difference of the first and last sample in the sub-ramp. For all the following sub-ramps the difference of the last two samples of the previous sub-ramp is compared to the last sample of the new sub-ramp.

If deviations are under a certain threshold, then no information of glitches is available.

3- Fit all safe readouts in a ramp with the currently implemented ramp-fitting algorithm till glitch information occurs. If a glitch occurs then reject the 8 following samples and go to point 1Average all fitted slopes when the whole ramp was processed.

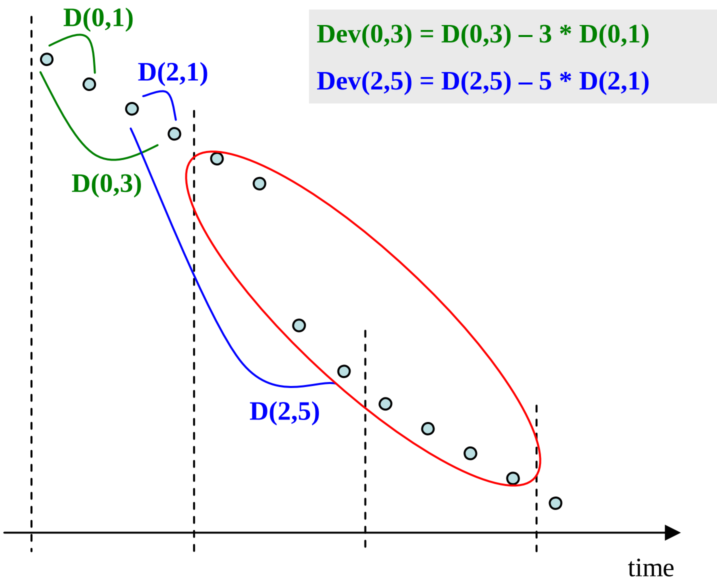


Figure 71. Example showing the crossed slope deviation detection

B.2.3.2 Ramp-Fitting

B.2.3.2.1 Least Squares Fitting

The least squares solution can be easily calculated in analytic form and it is optimal with respect to the Gaussian noise process. However, in case of heavy outliers (i.e., glitches) it performs very poor.

Figure 72(a) shows an example where least squares is performing very well, whereas *Figure 72*(b) shows the least squares solution on the same data as in *Figure 72*(a) where one sample is an outlier. One can clearly see that the obtained ramp is far from being perfect.

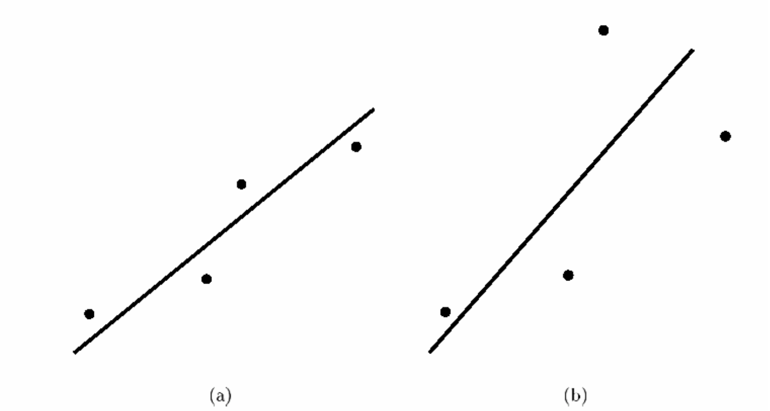


Figure 72. Least Square Fitting (1)

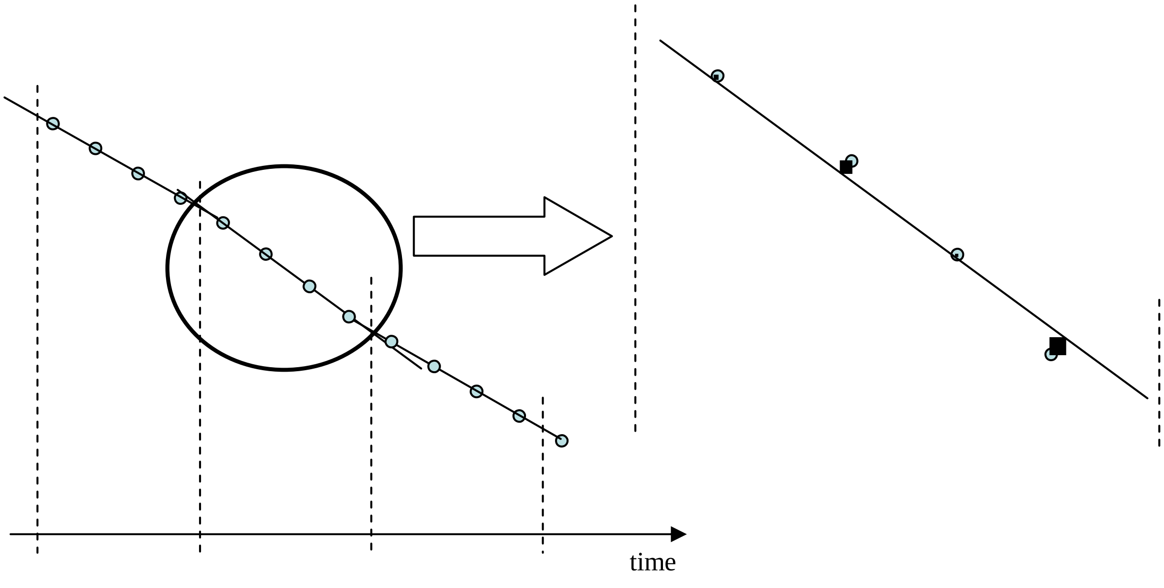


Figure 73. Least Square Fitting (2)

The slope of this fitting method is generated for a ramp or sub-ramps as follows

$$slope = \frac{\sum x \sum y - n\sum xy}{\left(\sum x\right)^2 - n\sum x^2}$$

where

$\sum x$:= sumX

$\sum x^2$:= Sum_of_X_Square

$\left(\sum x\right)^2$:= Sum_of_X_By_X

$\sum xy$:= sumXY

$\sum y$:= sumY

are float values. The result is an integer value where the float result is truncated.

Pseudo-Code example:

```
        /*=================================*/
       /* Least Square Tables Preparation */

        Sum_of_X[0]=Sum_of_X_Square[0]= Sum_of_X_By_X[0]=0;
        LSQ_Denominator[0]= 1; /* To Avoid Division By Zero */

        for(i=1;i<512;i++)
        {
                Sum_of_X[i] = i+Sum_of_X[i-1];
                Sum_of_X_Square[i] =(i*i)+Sum_of_X_Square[i-1];
                Sum_of_X_By_X[i]= Sum_of_X[i]*Sum_of_X[i];
                LSQ_Denominator[i]= ((i+1) * Sum_of_X_Square[i] - Sum_of_X_By_X[i]);
        }


Definitions :

                float Sum_of_X, Sum_of_X_By_X, Sum_of_X_Square, LSQ_Denominator;
                unsigned int i;


       /*==========================================*/
        /* Slope Calculation for Every Ramp/Sub-ramp */
        /* calculated in ramp_ft.c */
        /* denominator isset in write.c */


        for(i=0; i<ramps; i++)
        {
                sumY = sumXY = 0;
```

```
            /* Intermediate Sums */
            for(j=0; j<RRR; j++)
            {
                    sumXY += j*y[j];
                    sumY += y[j];
            }

            /* Intermediate Results */
            slope_z = (RRR * sumXY - sumY * sumX[RRR -1]);
            slope_n = denominator[RRR-1];

            if(slope_z>0) /* Choose the Rounding Error for the Division */
            /* Write The Final Result into Buffer */
                    MEMWRITE((u4*)(buffer+i+1), (int)((slope_z/slope_n)*RRR+0.5)&0xffff);
            else
            /* Write The Final Result into Buffer */
                    MEMWRITE((u4*)(buffer+i+1), (int)((slope_z/slope_n)*RRR-0.5)&0xffff);

            y += RRR;
    }


Definitions :

            float sumX, sumY, sumXY, denominator, slope_z, slope_n;
            int y;
            unsigned int i, j, RRR;
```

Example:

with RRR = **64**

```
 -----+------
    X      Y
  -----+------
    5   12457
    6   12509
    7   12563
    8   12616
  -----+------
```

`sumX` = **26.**

`Sum_of_X_Square` = **174.**

`Sum_of_X_By_X` = **676.**

`sumXY` = **326208.**

`sumY` = **50145.**

The numerator is:

`slope_z` = `sumX sumY – n sumXY` = 1303770. – 1304832. = **-1062.**

The denominator is:

`slope_n` = `Sum_of_X_By_X – n Sum_of_X_Square` = 676. – 696. = **-20.**

The result is therefore:

`slope` = (`slope_z/slope_n`)`*RRR±0.5` = **53.1*64-0.5** (float) => `slope` = **3397** (integer)

B.2.3.2.4 Two-Sample-Fit

This ramp-fitting method consists of calculating the slope by the difference of the second and last but one sample point. The advantage of this method is the simplicity of the difference value calculation. The disadvantage is that all other samples of the ramp are neglected. But in mind of a cumulative process where all information of the preceding samples are included in the last sample point this method

Assumptions on the points of the support set:

- No glitches (should be eliminated by the deglitching task)
- No non-linearity (e.g. saturation, etc.)

The calculation of the differences is performed as follows:

$$D(i) = X((i-1)*N + 2) - X(i*N - 1) \qquad \text{for } N \geq 4$$

The differences are represented as unsigned integer values. Figure 74 shows the ramp-fitting method.

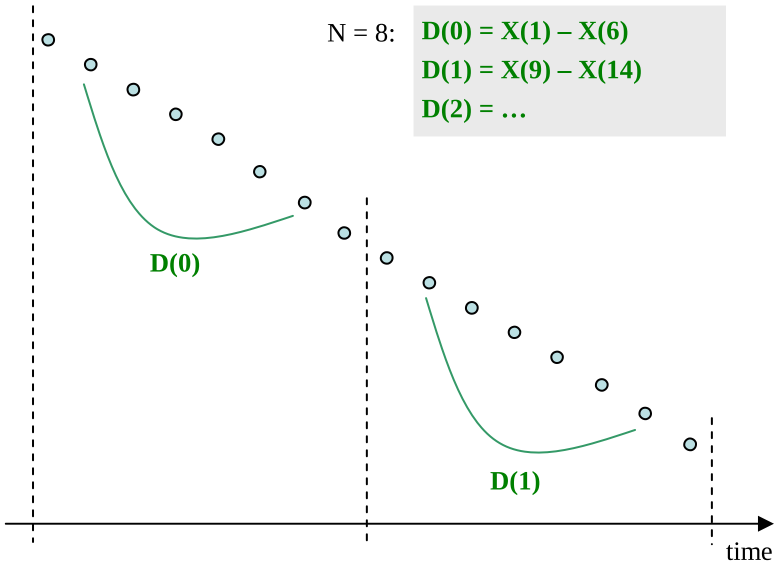


Figure 74. Example showing the ramp-fitting method (fit and residuals)

B.2.3.2.5 Slope Deviation Detection Method

The idea of the ramp fitting using the Slope Deviation Detection Method is very simple (*Figure 76* shows one example ramp using the Matlab generator with three glitches). It performs the following:

4- Calculate the differences between all successive readouts in a ramp. As the readout interval is equidistant, the differences represent the slopes using two successive readouts.

D(i) = X(i) – X(i+1) (Figure 77)

5- Calculate the deviation of these slopes. It is the differences of the differences of successive samples.

Dev(i) = D(i+1) –D(i) (Figure 78)

If no glitches occur, all deviations would be close to zero

6- Remove all deviations that are above 0.03 (3% of slope precision)

7- Average all the remaining slopes (This is the result)

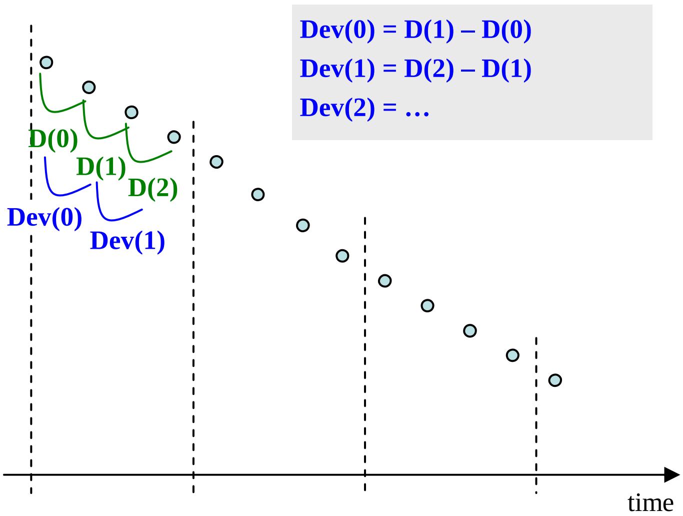


Figure 75 Example showing the slope deviation detection method

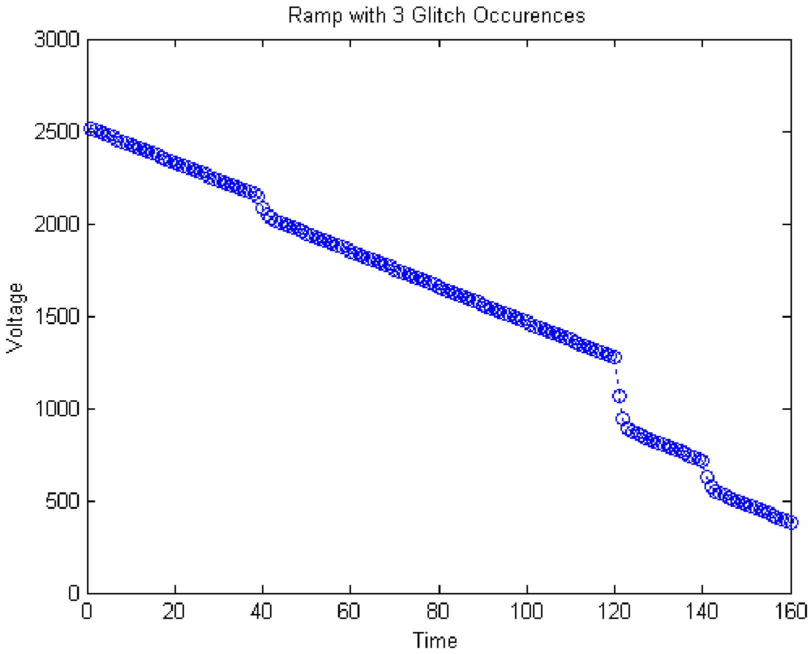


Figure 76. Photoconductor Readouts from 1 Channel with Additive Noise

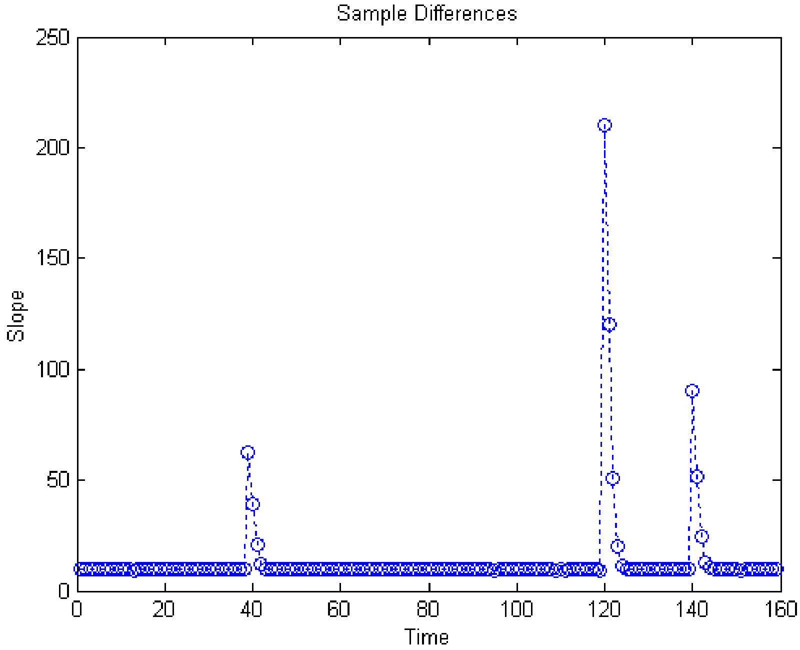


Figure 77. Successive Slopes in the Ramp

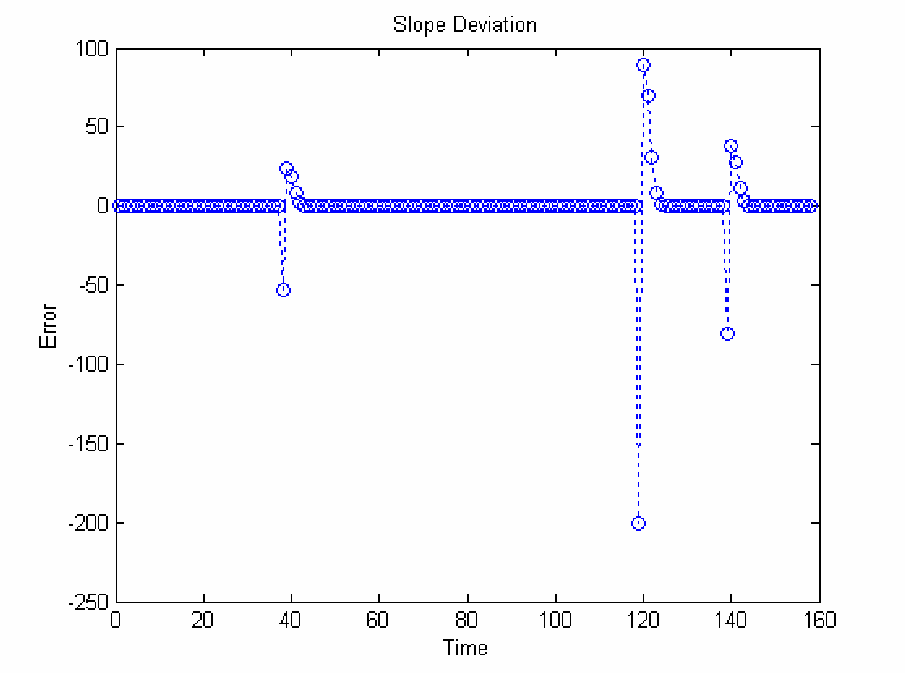


Figure 78. Successive Slope Deviations in the Ramp

Advantages of the Method

- Very Robust as all the slopes are tested
- Very fast it only calculates samples differences. For N readouts (3N-3) operations are required
- Well-suited for PACS. As each readout is equivalent to the number of photon/time. For an equidistant readout interval, this number is fix in the ideal case. It only changes with any temperature drift (Glitch), which will be easily detected and rejected.
- It could be generalized for other applications

Disadvantages of the Method

- Not suited to short ramps. If a glitch occurs, it would be hard to find the best slope for a long relaxation time (For PACS, we decided to have ramps no shorter to one second as the CRE are only stable and precise for a long ramp)

Conclusion

The method was tested for several ramp cases and compared with standard methods (RANSAC and Least Squares). It gives the best results with a very low complexity. We have also optimised the complexity of the algorithm to 3N/2-1 for the same precision.

B.2.3.2.6 Two-Samples Difference Method

In case of 2 samples to fit, the slope is calculated using the difference between two successive samples i.e. 32 slopes are calculated out of a 64 samples ramp

The calculation of the differences is performed as follows:

$$D(i) = X(2*i - 1) - X(2*i)$$

The differences are represented as unsigned integer values.

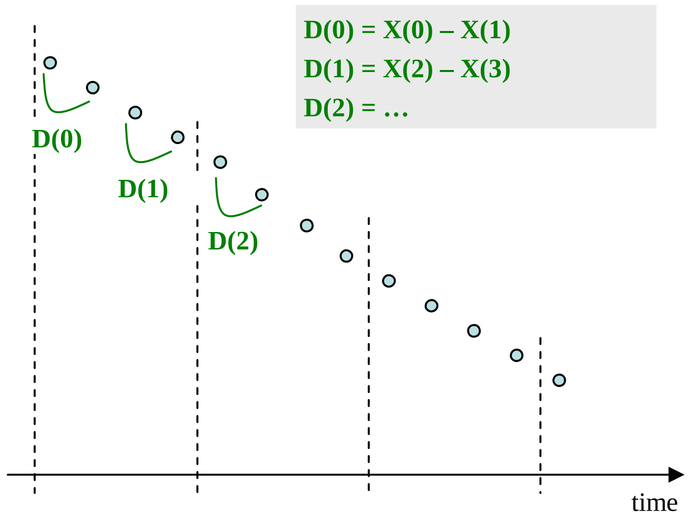


Figure 79. Example showing the Two-Samples Difference Method

B.2.3.2.7 Subramp Fitting

The SPU On-board SW reduces the ramp by deriving either the ramp slope or the ramp mean depending on the algorithm chosen by the the user. It contains an option to derive not only 1 slope/mean per reset interval, but also slopes/means for every 2,4,8,16,32,... samples within a ramp. The number of samples to fit is a commandable parameter. *Figure* 80 illustrates the sub-ramp fitting procedure. The fitting result is then subslope (in case of least square fit) of submean (in case of mean algorithm). The used algorithm could be read from the compressed entity header (See RD016 for more details).

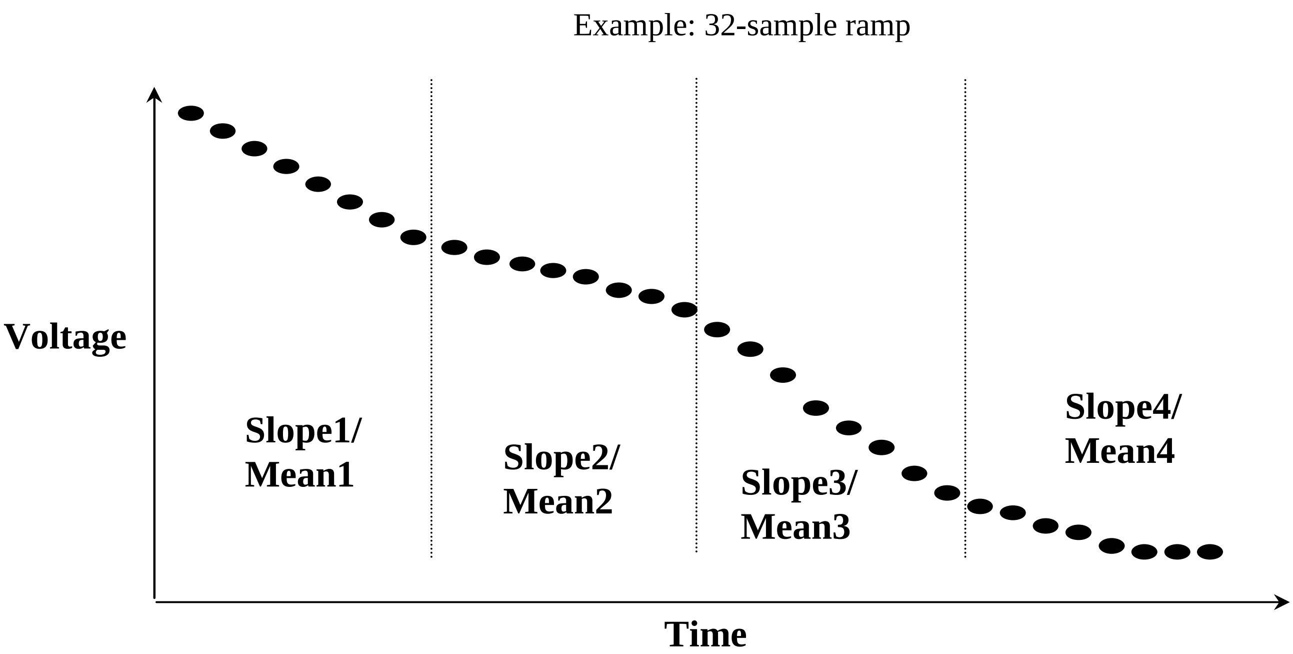


Figure 80. Illustration of Sub-ramp Fitting

B.2.4 Averaging and Rounding in Photometry

Averaging and rounding in photometry is now also done by the FM Averaging algorithm (see a few pages above). The "old" algorithm can still be used and is described hereafter.

Rounding has been developed and implemented as a countermeasure against high noise settings. In the default averaging process 4 samples are added up and divided by 4. The datatype is 16 bit short and no decimal digits are kept, so the value 2 has to be added before division to get a rounded average instead of a truncated one. In other words, there is already a 2-bit rounding made in the nominal averaging process. The SPU software is able to intensify the rounding inside the averaging process.

Instead of the default $mean = \frac{(y(x) + y(x+1) + y(x+2) + y(x+3) + 2)}{4}$ additional roundings can be

calculated by $mean = \mathrm{int}(\frac{(y(x) + y(x+1) + y(x+2) + y(x+3) + 4)}{8}) * 2$ and so on.

In addition to that, the number of samples to average is also freely configurable. Instead of 4 samples, 3, 5, or any other number 0<n<120 can be chosen.

For more information see document RD020.

B.3 Decimation

B.3.1 Decimation in Photometry

Beginning with SPU SW version 13.95 frame dropping is implemented via two uploadble parameters DPRE and DPOST (see user manual for usage). DPRE determines the number of frames to be discarded at the beginning of an averaging group and DPOST does so for the end. An averaging group is 4 frames in default mode, 8 frames in double mode or any number of frames when the number of frames to average is overridden with the NAVG parameter. The implementation is done by a preprocessing module (in p_proc.c) that copies (in place) frames to be averaged and leaves out frames to be dropped. The reduced number of frames and the modified number of frames to average are then passed to the averaging module.

B.3.2 Decimation in Spectroscopy

Similarly, frames at the start of a complete ramp and at its end can be dropped. The implementation is the same as in photometry with the difference that not the averaging groups (= the submeans) are decimated, but the whole ramp is.