474-00090
Effective Date: July 31, 2011
Revision -

GSFC JPSS CMO
September 24, 2011
Released

Joint Polar Satellite System (JPSS), Ground Project
Code 474
474-00071

Joint Polar Satellite System (JPSS) Operational Algorithm Description Document for Cross-Track Infrared Sounder (CrIS) Sensor Data Record (SDR)

For Public Release

The information provided herein does not contain technical data as defined in the International Traffic in Arms Regulations (ITAR) 22 CFC 120.10. This document has been approved For Public Release to the NOAA Comprehensive Large Array-data Stewardship System (CLASS).

Goddard Space Flight Center
Greenbelt, Maryland

National Aeronautics and
Space Administration

This page intentionally left blank.

Joint Polar Satellite System (JPSS) Operational Algorithm Description Document for Cross-Track Infrared Sounder (CrIS) Sensor Data Record (SDR)

JPSS Electronic Signature Page

Reviewed By:

Neal Baker
JPSS Data Products and Algorithms, Senior Engineering Advisor
(Electronic Approvals available online at https://jpssmis.gsfc.nasa.gov/mainmenu_dsp.cfm)

Approved By:

Heather Kilcoyne
DPA Manager
(Electronic Approvals available online at https://jpssmis.gsfc.nasa.gov/mainmenu_dsp.cfm)

Goddard Space Flight Center
Greenbelt, Maryland

This page intentionally left blank.

Preface

This document is under JPSS Ground Algorithm ERB configuration control. Once this document is approved, JPSS approved changes are handled in accordance with Class I and Class II change control requirements as described in the JPSS Configuration Management Procedures, and changes to this document shall be made by complete revision.

Any questions should be addressed to:

JPSS Configuration Management Office
NASA/GSFC
Code 474
Greenbelt, MD 20771

This page intentionally left blank.

Abstract

The purpose of the Operational Algorithm Description (OAD) document is to express, in computer-science terms, the remote sensing algorithms that produce the National Polar-Orbiting Operational Environmental Satellite System (NPOESS) end-user data products. These products are individually known as Raw Data Records (RDRs), Temperature Data Records (TDRs), Sensor Data Records (SDRs) and Environmental Data Records (EDRs). In addition, any Intermediate Products (IPs) produced in the process are also described in the OAD.

The scope of this document is limited to the description of the core operational algorithm required to create the CrIS SDR.

Keywords: JPSS, IP, SDR, OAD, CrIS

This page intentionally left blank.

Change History Log

Revision	Effective Date	Description of Changes (Reference the CCR & CCB/ERB Approve Date)
Original	06/03/2011	This version incorporates **474-CCR-11-0087**: (This version baselines D39132, Rev B dated 07/07/2010. This is the version that was approved for NPP launch. Per NPOESS CDFCB - External, Volume V – Metadata, doc number D34862-05, this has been approved for Public Release into CLASS.) to create JPSS version, Rev-. This was approved by the JPSS Ground Algorithm ERB on June 3, 2011.

This page intentionally left blank.

Doc No: D39132
Rev B
Date: 07 Jul 2010
Subcontract No. 7600002744

NATIONAL POLAR-ORBITING OPERATIONAL ENVIRONMENTAL SATELLITE SYSTEM (NPOESS)

OPERATIONAL ALGORITHM DESCRIPTION DOCUMENT FOR CROSS-TRACK INFRARED SOUNDER (CrIS) SENSOR DATA RECORD (SDR)

SDRL No. S141
SYSTEM SPECIFICATION SS22-0096

RAYTHEON COMPANY
INTELLIGENCE AND INFORMATION SYSTEMS (IIS)
NPOESS PROGRAM
OMAHA, NEBRASKA

IAW DFAR 252.227-7036, Raytheon hereby declares that, to the best of its knowledge and belief, the technical data delivered under Subcontract No. 7600002744 is complete, accurate, and complies with all requirements of the Subcontract.

TITLE: NATIONAL POLAR-ORBITING OPERATIONAL ENVIRONMENTAL SATELLITE SYSTEM (NPOESS) OPERATIONAL ALGORITHM DESCRIPTION DOCUMENT FOR CROSS-TRACK INFRARED SOUNDER (CrIS) SENSOR DATA RECORD (SDR)

APPROVAL SIGNATURES:

______________________________ 07 Jul 2010
Stephen E. Ellefson Date
ING/PRO Lead

______________________________ 07 Jul 2010
Gabriela A. Ostler Date
Mission Assurance and Enterprise Effectiveness (MAEE)

Northrop Grumman Space & Mission Systems Corp.
Space Technology
One Space Park
Redondo Beach, CA 90278

Engineering & Manufacturing Development (EMD) Phase
Acquisitions &Operations Contract

CAGE NO. 11982

Operational Algorithm Description
CrIS SDR

Document Number: D39132

Document Date: Jul 07, 2010 **Revision: B**

PREPARED BY:

Degui Gu *Models & Simulations Lead*	*Date*	Paul D. Siebels *IDPS Processing SI Software Manager*	07 Jul 2010 *Date*

ELECTRONIC APPROVAL SIGNATURES:

Roy Tsugawa *SEIT Lead & ACCB Chair*	*Date*	Bob Hughes *Algorithm Implementation Thread Lead*	*Date*
Bob Hughes *Data Product System Engineering Lead*	*Date*	Stephen E. Ellefson *IDPS Processing SI Lead*	07 Jul 2010 *Date*

Northrop Grumman Space & Mission Systems Corp.
Space Technology
One Space Park
Redondo Beach, CA 90278

Prepared for
Department of the Air Force
NPOESS Integrated Program Office
C/O SMC/CIK
2420 Vela Way, Suite 1467-A8
Los Angeles AFB, CA 90245-4659

Under
Contract No. F04701-02-C-0502

This document has been identified per the NPOESS Common Data Format Control Book – External Volume 5 Metadata, D34862-05, Appendix B as a document to be provided to the NOAA Comprehensive Large Array-data Stewardship System (CLASS) via the delivery of NPOESS Document Release Packages to CLASS.

Northrop Grumman Space & Mission Systems Corp. **Space Technology** One Space Park Redondo Beach, CA 90278	NORTHROP GRUMMAN Raytheon
Revision/Change Record	**Document No.** **D39132**

Revision	Document Date	Revision/Change Description	Pages Affected
---	6-17-04	Initial version.	N/A
A1	6-30-04	Modifications to include references to Bomem documentation monitored and tunable parameters, additional detail of source code classes, specific dependence on commercial libraries.	2, 7-10, 12-14, 18-23
A2	6-30-04	Sec. 2.1.1.1 updated to reflect ATBD --- included: 1. CrIS Specific Calibration Equation 2. Geolocation 3. FFT Sec. 2.2.1 Table 4 updated to reflect configuration file. Sec. 2.2.2 Outputs discussing apodization and EDR inputs. Sec. 2.2.2 Table 8 updated with units. Sec. 5.1 Tables 12 and 13 updated to reflect paths in VTP.	4-6 9-11 11-12 14-17 33-37
A3	6-30-04	Invalid Data SDR QF is set if any of the other error flags (except Excess Thermal Drift SDR Quality Flag) are set.	29
A4	6-30-04	Table 13 updated to reflect list of delivered data.	36-37
A	10-26-04	Sec 4.4 Added two new SDR quality flags: SAA and Lunar Inception. Sec 2.2.2 Added guideline for converting SDR outputs into EDR inputs. Sec 2.1.1.1 Corrected an error in the implemented calibration equation: F int = 1, was F int = F int –1 = 1.	30-31 12-13 6

Northrop Grumman Space & Mission Systems Corp. **Space Technology** One Space Park Redondo Beach, CA 90278	NORTHROP GRUMMAN Raytheon
Revision/Change Record	**Document No.** **D39132**

Revision	Document Date	Revision/Change Description	Pages Affected
B1	6-18-06	21Nov05 – Reflects Raytheon-Omaha's Science To Operational Code Conversion edits, but **does not include** making 19Oct05 CrIS SDR Tech Memo changes about Fringe Count Error (FCE) Handling or any Quality Flag follow-on updates, although Omaha added a "Value" column to Table 4.4-2, SDR Quality Flags. 01Mar06 – Removed NPOESS-Omaha's electronic signatures from title/signature page, changed copyright to 2006 on coversheet, put current dates on upper right header. Title/Signature page, and Revision Change/History page. Replaced initial Sci2Ops Code Completion 120-page Unit Test dated 10/28/05 with the follow-on 10-page Unit Test dated 02/28/06 to address Fringe Count Errors, then updated Table of Content/List of Figures/List of Tables/TBD page numbers. 21Mar06 – Changed Section 2.4, Graceful Degradation to Section 2.7 and inserted Sections 2.4, 2.4.1, 2.4.2, 2.4.3, 2.5, and 2.6 discussing the "Sliding Window" concept. 04Apr06 – Deleted a couple of "Algorithm Control Parameters" from Table 2.2-5. Deleted a couple of words from the table under Section 2.2.2, Outputs. 12Apr06 – Worked off the 30Mar06 follow-on CCPR comments to include the following changes. Replaced Figure 2.2-1, IPO Model Interface to INF & DMS, with the latest version. Removed Section 2.1.2, Prototype Code. Removed page 6 paragraph discussing Fringe Count Errors. Deleted Section 2.1.1.1 words discussing "Sensor 8 Second Scan Epoch Timing" to include Figure 2.1-1 about this subject. Replaced Tables 2.2-2, 2.2-3, and 2.2-4. 17Apr06 – Inserted current Unit Test dated 13Apr06 and removed obsolete one. Added 21Mar06, 04Apr06, and 12Apr06 comments to the Revision/change Record page to log document edits. Made spacing/format edits. Updated TBD/TBR page. Updated List of Figures to include Figures 2.4-1 and 2.6-1. Updated List of Tables' page numbers. Added Raytheon-Omaha electronic signatures in preparation for official delivery of document to prime. 18Apr06 – Did minor spacing/format edits. Put two character spaces between sentences. Changed a few text paragraphs from 12 pt or 10 pt to be consistent so they were all Arial 11 pt throughout the document.	All
B2	6-27-07	Updated quality flags per Tech Memo NP-EMD-2007.510.0027.Rev.A . Updated logo, header, dates, etc.	All
B3	7-10-07	Delivered to NGST.	All
B4	12-17-07	ECR A-103, EDR-PR 1.8 CP 3, and CDFCB-X compliance updates – Added new output structure definitions for CrIS-SDR and CrIS-SDR-GEO.	All

Northrop Grumman Space & Mission Systems Corp. **Space Technology** One Space Park Redondo Beach, CA 90278	NORTHROP GRUMMAN Raytheon
Revision/Change Record	**Document No.** **D39132**

Revision	Document Date	Revision/Change Description	Pages Affected
B5	2-28-08	ISTN_CRIS_SDR_UNIX_NGST_1.1 science drop and TM MS_Tech_ Memo_CrIS_SDR_QF_Updates_RevB , NP-EMD-2007.510.0027.Rev.B, implementation. Format updates for CDFCB-X compliance. 1. Table 7 – added 6 new configurable parameters 2. Table 9 – format updates 3. Tables 25 through 29 – quality flag updates 4. Paragraph 1.3.2 – added the above-mentioned TM to the NPOESS Reference Documents table; changed reference to CDFCB Vol I to ref. Vol III, instead Responses to comments provided in 070710 SDRL S141 D39132_B3_CrIS_SDR_OAD_review 1. Figure 4 – updated Figure with explanations 2. Paragraph 2.1.2.9 – provided clarification on the meaning of “hard-coded the sliding window size” 3. Paragraph 2.1.2.9.2 – provided explanations for the variables W and S in the two equations 4. Paragraph 2.1.2.9.1 - Updated the paragraph to use the phrase “degraded quality” instead of “invalid”.	2, 11, 13, 15, 24-25, 31-33, 36
B6	4-11-08	Added Cris SDR Exit Angles IP to Table 8 per ECR R871. Implemented Tech Memos NP-EMD.2005.510.0118 and NP-EMD.2005.510.0130. Reformatted in accordance to new template. Addressed Peer Review comments. Delivered to NGST.	All
B7	4-17-08	Updated description of the CrIS SDR Exit Angles IP in Table 8. Updated paragraph 2.1.2.2.1 (ScienceDataProcessor Attributes) with an explanation between the ATBD and operational software’s Planck Radiation Constants.	13, 19
B8	3-03-09	NP-EMD.2008.510.0055 Rev A, PCR019301: Updates after Sensor Characterization Phase I: 1. Add CrIS SDR ATBD (D43773) Rev A in Table 1 Document References. 2. Add CrIS SDR operational software in Table 2 Source Code References. 3. Change ICT Emissivity Constant to ICT Emissivity in Table 6 Tunable Parameters Provided via the Four Minute Engineering Telemetry RDR. 4. Add a function generateUSN in Table 18 Spectra Operations.	2,3,8,22
B9	3-06-09	Prepared for SDRL Delivery (for Sensor Characterization)	All
B10	3-10-09	Redelivered to NGAS with Template corrections	1-37
B11	12-01-09	Updated for RFA No.551 and updated subcontract number	Title pg & Table 9

Northrop Grumman Space & Mission Systems Corp.
Space Technology
One Space Park
Redondo Beach, CA 90278

NORTHROP GRUMMAN

Raytheon

Revision/Change Record

Document No.
D39132

Revision	Document Date	Revision/Change Description	Pages Affected
B12	12-17-09	Added six new classes (CalibrationSpectra, CCSDSFormat, CorrectionMatrix, EngineeringCalibrationRecord, Interferogram and TemeratureHistory) and updated six original classes (ScienceDataProcessor, SpectraManager, Spectra, TelemetryProcessor, VideData and ScienceCalibrationRecord) Delete CrIS Specific Calibration Equation in Section 2.1.2.13 Add the following tunable parameters in Table 6: linearityCorrectionA2Parameters lwLinearityCorrectionParameter_a2 mwLinearityCorrectionParameter_a2 swLinearityCorrectionParameter_a2 linearityCorrectionVinstParameters lwLinearityCorrectionParameter_Vinst mwLinearityCorrectionParameter_Vinst swLinearityCorrectionParameter_Vinst linearityCorrectionModEffParameters lwLinearityCorrectionParameter_ModEff mwLinearityCorrectionParameter_ModEff swLinearityCorrectionParameter_ModEff lwFirAccumulatorStartBit mwFirAccumulatorStartBit swFirAccumulatorStartBit	19-37 43

Revision	Document Date	Revision/Change Description	Pages Affected
		lwFirFilterScale mwFirFilterScale swFirFilterScale lwPassBandStar mwPassBandStart swPassBandStart lwPassBandStop mwPassBandStop swPassBandStop firFilterResponse Longwave Real Imag Midwave Real Imag Shortwave Real Imag GainMappingLW GainMappingMW GainMappingSW Add the following tunable parameters in Table 7: ShiftFactorArray AppShiftFactorFlag performLWLinearityCorrectionControl performMMLinearityCorrectionControl performSWLinearityCorrectionControl lwLinearityCorrectionControlParam lwLinearityCorrectionControlParam lwLinearityCorrectionControlParam	
B13	2-23-10	Incorporated CCPR comments and prepared for SDRL delivery	All
B14	3-24-10	Implemented TM 2009.510.0046	All
B15	5-20-2010	Updated Table 3 with Two Line Element (TLE) listed as a required algorithm input and Table 8 with CrIS Correction Matrix internal auxiliary item listed as an optional output of the algorithm.	7,18
B	7-07-10	Prepared for TIM/ARB/ACCB	All

Table of Contents

List of Figures

List of Tables

1.0 INTRODUCTION

1.1 Objective

The purpose of the Operational Algorithm Description (OAD) document is to express, in computer-science terms, the remote sensing algorithms that produce the National Polar-Orbiting Operational Environmental Satellite System (NPOESS) end-user data products. These products are individually known as Raw Data Records (RDRs), Temperature Data Records (TDRs), Sensor Data Records (SDRs) and Environmental Data Records (EDRs). In addition, any Intermediate Products (IPs) produced in the process are also described in the OAD.

The science basis of an algorithm is described in a corresponding Algorithm Theoretical Basis Document (ATBD). The OAD provides a software description of that science as implemented in the operational ground system -- the Data Processing Element (DPE).

The purpose of an OAD is two-fold:

1. Provide initial implementation design guidance to the operational software developer.
2. Capture the “as-built” operational implementation of the algorithm reflecting any changes needed to meet operational performance/design requirements.

An individual OAD document describes one or more algorithms used in the production of one or more data products. There is a general, but not strict, one-to-one correspondence between OAD and ATBD documents. This particular document describes operational software implementation for the Cross-track Infrared Sounder (CrIS) Sensor Data Record (SDR).

1.2 Scope

The scope of this document is limited to the description of the core operational algorithm required to create the CrIS SDR. The theoretical basis for this algorithm is described in the Algorithm Theoretical Basis Document for the Cross Track Infrared Sounder (CrIS), Volume I, Sensor Data Records (SDRs), D43773.

1.3 References

1.3.1 Document References

The science and system engineering documents relevant to the algorithms described in this OAD are listed in Table 1.

Table 1. Document References

Document Title	Document Number/Revision	Revision Date
NPP EDR Production Report	D37005 Rev. D	11 Feb 2009
EDR Interdependency Report	D36385 Rev. F	19 May 2009
CrIS Sensor Calibration and Validation Master Plan (ITT Document)	8180003 Rev. D	23 Feb 2005
NPP Mission Data Format Control Book (MDFCB)	D48190-01 Rev. B	03 Sep 2009
SDR Generator Validation Test Procedure	VTP8189758 Rev. D	12 Sep 2005
SDR Generator Software User Manual	UM8189758	Latest Revision Date
SDR Generator Rose Model	RM8189758	Latest Revision Date
CrIS_Science_Data_Packet_Format_Oct2003_v3.xls	Rev. V3	23 Oct 2003
CrIS Command and Data Packet Dictionary	8196185 Rev. H	14 Nov 2008
Data Processor Inter-Subsystem Interface Control Document (DPIS ICD)	D35850 Rev. AA	12 May 10
D35836_H_NPOESS_Glossary	D35836 Rev. H	03 Mar 2009
D35838_H_NPOESS_Acronyms	D35838 Rev. H	03 Mar 2009
CDFCB-X Volume I - Overview	D34862-01 Rev F	08 Dec 2009
CDFCB-X Volume II – RDR Formats	D34862-02 Rev. D	03 Jun 2009
CDFCB-X Volume III – SDR/TDR Formats	D34862-03 Rev. F	16 Apr 2010
CDFCB-X Volume IV Part 1 – IP/ARP/GEO Formats	D34862-04-01 Rev. F	16 Apr 2010
CDFCB-X Volume IV Part 2 – Atmospheric, Clouds, and Imagery EDRs	D34862-04-02 Rev. F	16 Apr 2010
CDFCB-X Volume IV Part 3 – Land and Ocean/Water EDRs	D34862-04-03 Rev. F	16 Apr 2010
CDFCB-X Volume IV Part 4 – Earth Radiation Budget EDRs	D34862-04-04 Rev. F	16 Apr 2010
CDFCB-X Volume V - Metadata	D34862-05 Rev. F	09 Dec 2009
CDFCB-X Volume VI – Ancillary Data, AuxiliaryData, Reports, and Messages	D34862-06 Rev. J	21 May 2010
CDFCB-X Volume VII – NPOESS Downlink Formats	D34862-07 Rev. C	08 Dec 2009
CDFCB-X Volume VIII – Look Up Table Formats	D34862-08 Rev. D	16 Apr 2010
MS_Tech_ Memo_CrIS_SDR_QF_Updates_RevB	NP-EMD-2007.510.0027 Rev. B	10 Oct 2007
MS Tech Memo CrIS SDR QF Updates_0027_Rev_A	NP-EMD-2007.510.0027 Rev. A	07 Oct 2007
Memo_CrIS_SDR_OAD_Update-v2	NP-EMD.2005.510.0118	09 Sep 2005
Memo_CrIS_SDR_OAD_Revision	NP-EMD.2005.510.0130	19 Oct 2005
D41869_(latest revision)_ VIIRS_CmnGeo_OAD.doc.	D41869 / latest rev	18 Mar 2009
Cross Track Infrared Sounder (CrIS) Sensor Data Records (SDR) Algorithm Theoretical Basis Document ATBD (ref BOM-CrIS-0067/ITT8180004)	D43773 Rev. D	17 Sep 2009
A&DP Tech memo CrIS_SDR OAD Update for CrIS_SDR_UNIX_1.1	NP-EMD.2007.510.0061	10 Oct 2007
A&DP Tech memo CrIS SDR OAD Updates for CrIS_SDR_UNIX_2.1	NP-EMD.2008.510.0055_RevA	20 Jan 2009
A&DP Tech memo CrIS SDR OAD Updates for CrIS-RevA__SDR_UNIX_2.2.doc	NP-EMD.2009.510.0045.Rev.A	22 Sep 2009
A&DP Tech memo TM_CrIS_Scan Baffle Temperature Correction	NP-EMD.2009.510.0046	16 Sep 2009
A&DP Tech memo TM_CrIS_SDR_Code_Updates	NP-EMD.2010.510.0006.RevA	26 Apr 2010

1.3.2 Source Code References

The science and operational code and associated documentation relevant to the algorithms described in this OAD are listed in Table 2.

Table 2. Source Code References

Reference Title	Reference Tag/Revision	Reference Date
SASS_CRIS_SDR_v 3.0.0 QL		
ISTN_CRIS_SDR_2.1	(OAD Rev ---)	16 Aug 2004
CrIS SDR operational software	B1.3	xx yyy 2004
ISTN_CRIS_SDR_NGST_2.1.1	ECR-A40 (OAD Rev 6)	04 Nov 2004
ISTN_CRIS_SDR_NGST_2.1.2	ECR-A065 (OAD Rev A)	06 Oct 2005
Memo_CrIS_SDR_OAD_Update-v2	NP-EMD.2005.510.0118	09 Sep 2005
Memo_CrIS_SDR_OAD_Revision	NP-EMD.2005.510.0130	19 Oct 2005
CrIS SDR science-grade software	ISTN_CRIS_SDR_NGST_UNIX_1.1 ECR-A135	18 Dec 2007
MS_Tech_ Memo_CrIS_SDR_QF_Updates_RevB	NP-EMD-2007.510.0027 Rev. B	10 Oct 2007
OAD Update	NP-EMD-2007.510.0061	
CrIS SDR operational software	B1.5 (OAD Rev B4)	17 Dec 2007
CrIS SDR science-grade software	ISTN_CRIS_SDR_NGST_UNIX_2.1 ECR-A175A	17 Dec 2008
CrIS SDR operational software	B1.5.x.1 (OAD Rev B6)	01 April 2008
CrIS SDR operational software (PCR019301)	IDPS Post-X, Build A (OAD Rev B9)	19 Feb 2009
CrIS SDR science-grade software	ISTN_CRIS_SDR_NGST_UNIX_2.2 ECR-A255 (OAD Rev B11)	09 Nov 2009
Tech Memo: CrIS SDR OAD Updates for CrIS-RevA__SDR_UNIX_2.2	NP-EMD.2009.510.0045.Rev.A	22 Sep 2009
CrIS SDR operational software	IDPS Sensor Characterization Build SC-7 (OAD Rev B12)	15 Feb 2010
OAD SDRL (no code changes)	OAD Rev B13	23 Feb 2010
CrIS SDR Algorithms Unix 2.2.1 Delta Delivery	ISTN_CRIS_SDR_UNIX_NGST_2.2.1(ECR-A275)	22 Mar 2010
CrIS SDR operational software(PCR21464) NP-EMD.20909.510.0049	IDPS Sensor Characterization Build SC-9 (OAD Rev B14)	24 Mar 2010
A&DP Tech memo TM_CrIS_SDR_Code_Updates NP-EMD.2010.510.0006.RevA	ISTN_CRIS_SDR_UNIX_NGST_2.2.2	29 Apr 2010
CrIS SDR operational software	IDPS Sensor Characterization Build SC-10 (OAD Rev B15)	25 May 2010
ACCB	OAD Rev B	07 Jul 2010

2.0 ALGORITHM OVERVIEW

This document presents the theoretical basis of the CrIS SDR Algorithms. The functional flow of algorithms required to transform a Raw Data Record (RDR) coming from the satellite into a Sensor Data Record (SDR) is described. These SDRs are then transformed into Environmental Data Records (EDRs). Physical retrievals of atmospheric parameters from infrared spectra are computed by accurate radiative transfer models, known as forward models, relating the atmospheric parameters to the observed channel radiances. The CrIS forward model is described in Section 4.2.2 Overview of the OSS Method and Section 4.2.3 Implementation of the OSS method, CrIS SDR ATBD (D43773), is not presented here.

This document describes the CrIS SDR Algorithms specific to processing required at the ground segment. It covers the processing needs for all data being sent to the ground when the instrument is operational, including observational and calibration data, for all measurements performed by the instrument. The algorithms for decoding and calibrating the calibration data (e.g. generation of Internal Calibration Target (ICT) radiance) are also covered here.

However, the present document does not cover the data manipulation related to all instruments of the NPOESS platform. In other words, it is assumed that the data entering the SDR algorithm processing chain is identical to the Consultative Committee on Space Data Systems (CCSDS) formatted data leaving the instrument on board. The processing of the data produced when the instrument is under test or characterization, e.g. during the Commissioning Phase, is excluded. The operational and processing steps required during the Commissioning Phase are addressed in the CrIS Calibration/Validation Master Plan (International Telephone & Telegraph (ITT) document 8180003). This identifies how CrIS is to be calibrated, what the baseline operational scenario is, how this scenario can be verified, etc. Figure 1 identifies the top level processing flow of the CrIS SDR Algorithm.

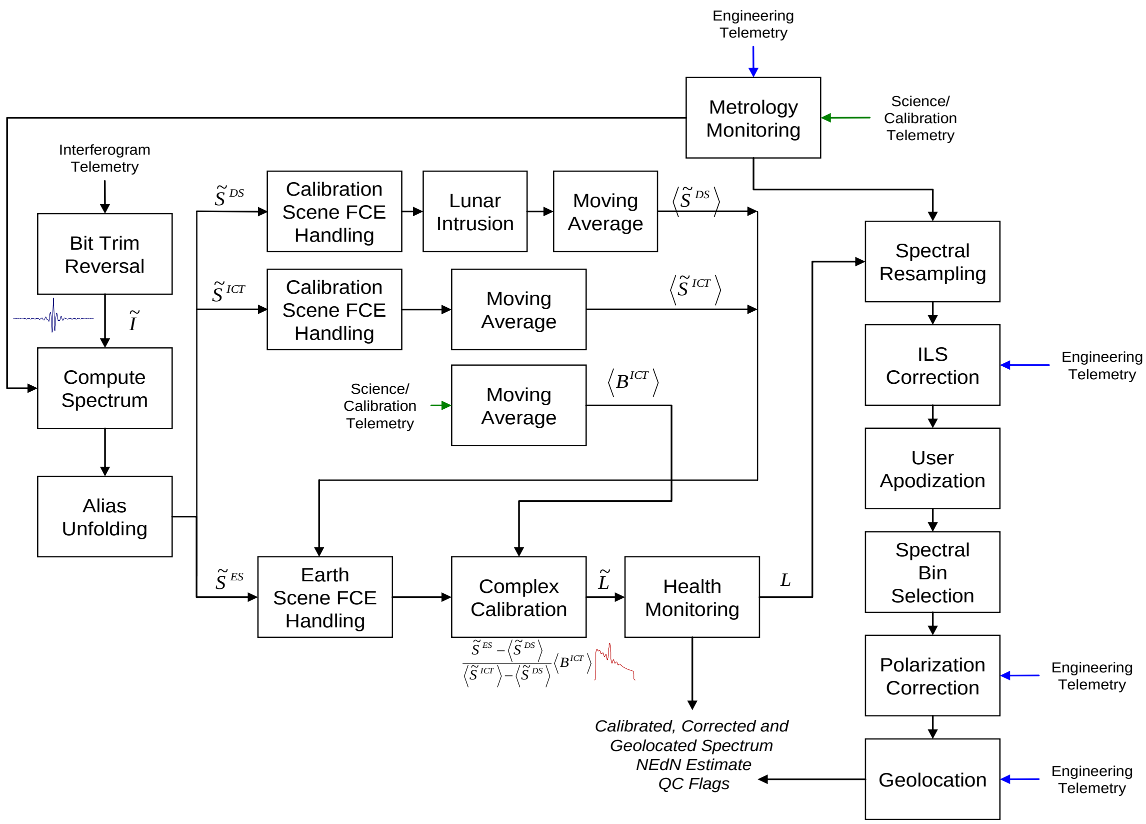

Figure 1. Processing Chain

2.1 CrIS SDR Description

2.1.1 Interfaces

To begin data processing, the Infrastructure (INF) Subsystem Software Item (SI) initiates the CrIS SDR algorithm. The INF SI provides tasking information to the algorithm indicating which granule to process. The Data Management Subsystem (DMS) SI provides data storage and retrieval capability. CrIS SDR processing is retaskable, so instead of shutting down after processing it requests additional tasking information from INF and continues processing with this information. A library of C++ classes implements the SI interfaces, as depicted in Figure 2.

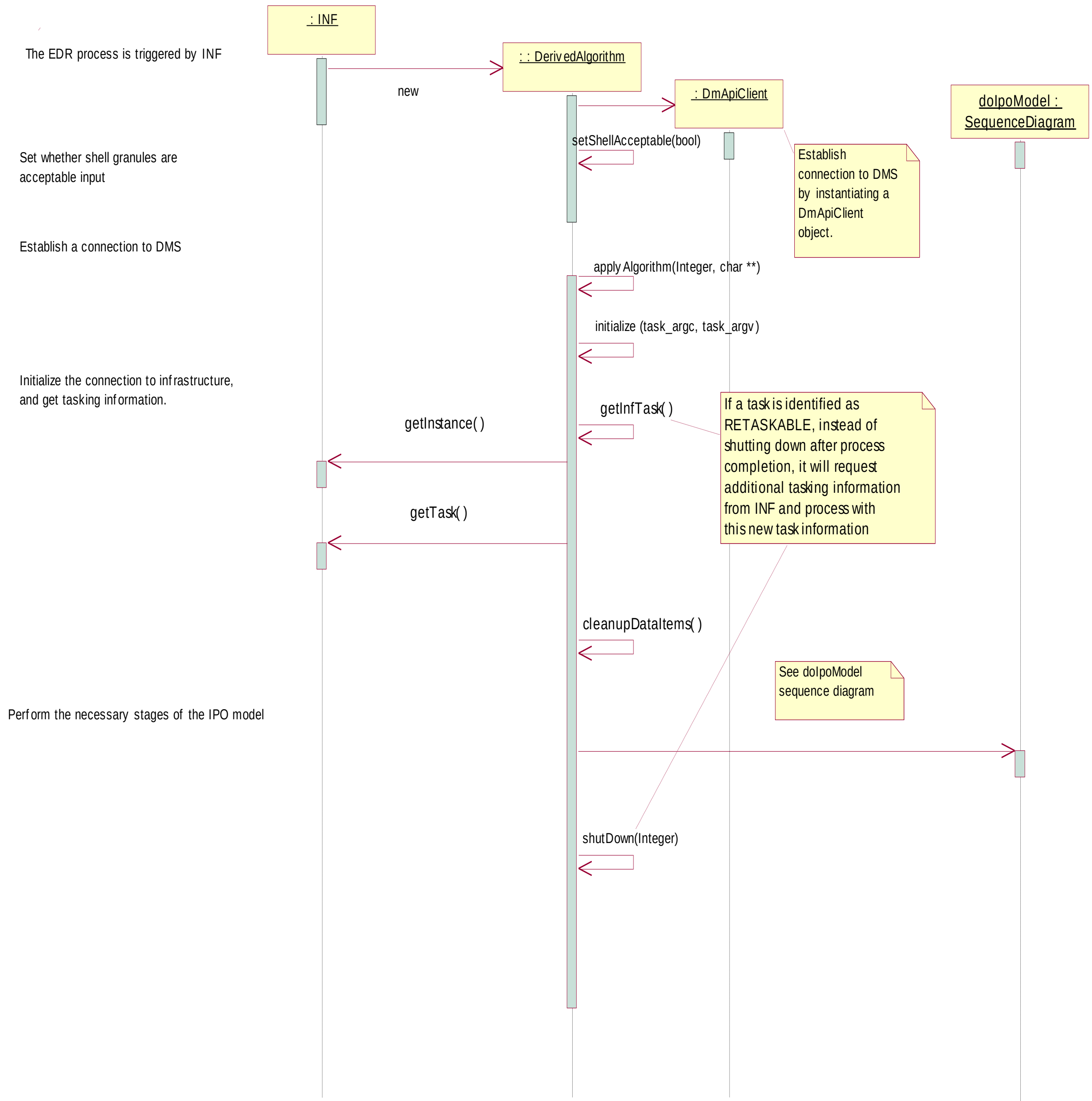

Figure 2. IPO Model Interface to INF and DMS

2.1.1.1 Inputs

Implementation of CrIS SDR requires CrIS RDRs, Space Craft Diary RDR, engineering calibration record, the correction matrix, and the tunable parameters. General description of the inputs used for CrIS SDR can be found in Table 3. Refer to the CDFCB-X, D34862, for a detailed description of the inputs.

Table 3. CrIS SDR Algorithm Inputs

Input	Description
CrIS RDR Data	Data Ingest stores in the DMS
Spacecraft Diary RDR	Ephemeris and Attitude Data
Correction matrix data / Engineering calibration record	Correction matrix last calculated to speed up processing and also the last received engineering calibration record
CrIS Configurable Parameters	Tunable parameters for the CrIS SDR algorithm
CrIS DQTT	Data quality threshold table
Two Line Element (TLE)	14 parameters calculated and provided by C3S to IDPS describing the orbit of the spacecraft as described on the NORAD website http://www.celestrak.com/NORAD/documentation/tle-fmt.asp
CrIS Fill Packet LUT	Lookup table which contains templates of each of the CrIS Earth Scene (ES), Deep Space (DS), and Internal Calibration Target (ICT) interferogram packets that are used as “fill” packets to replace packets missing from the CrIS RDR inputs.

The inputs to the CrIS SDR Algorithm via the CrIS RDR input consist of three types of CCSDS packets produced by the CrIS sensor. These include:

1. Interferogram RDR
    a. Internal Calibration Target (ICT) Interferogram RDR
    b. Deep Space (DS) Interferogram RDR
    c. Earth Scene (ES) Interferogram RDR
2. Four Minute Engineering Telemetry RDR
3. Eight Second Science/Calibration Telemetry RDR

The contents and formats of these packets are defined in the CrIS Command and Data Dictionary.

Tables 4, 5, and 6 identify the parameters within the Eight Second Science/Calibration and Four Minute Engineering Telemetry RDRs that are monitored throughout processing. Additional details of each of these packets and their associated content are provided in the MDFCB, GSFC 429-05-02-42.

Table 4. Parameters Monitored via the Eight Second Science/Calibration Telemetry RDR

Parameter	Comments	Units
IE CCA Calibration Resistor Temperature (epoch 1..40) - (refers to 200 msec epoch)	40 Readings	°C
Low Range Calibration Resistor (epoch 1..40)	40 Readings	°C
High Range Calibration Resistor (epoch 1..40)	40 Readings	°C
ICT Temperature #1 (epoch 1..40)	40 Readings	°C
ICT Temperature #2 (epoch 1..40)	40 Readings	°C
Cross Track Servo Error (sample 21, epoch 4..33)	30 Readings	N/A
In Track Servo Error (sample 21, epoch 4..33)	30 Readings	N/A
Laser Diode Current (epoch 39)	Single Reading	mAmps
Laser Diode Temperature (epoch 39)	Single Reading	°C
Beamsplitter Temperature #1 (epoch 39)	Single Reading	°C
OMA Structure Input Temperature #1 (epoch 39)	Single Reading	N/A
OMA Structure Input Temperature #2 (epoch 39)	Single Reading	°C
SSM Scan Mirror Temperature (epoch 39)	Single Reading	N/A
SSM Scan Mirror Baffle Temperature (epoch 39)	Single Reading	°K
Stage 2 Cooler Temperature (epoch 39)	Single Reading	°K
Stage 4 Cooler Temperature (epoch 39)	Single Reading	°K

Parameter	Comments	Units
Stage 1 Cooler Temperature (epoch 39)	Single Reading	°K
Stage 3 Cooler Temperature (epoch 39)	Single Reading	°K
Telescope Temperature #1 (epoch 39)	Single Reading	°C

Table 5. Parameters Monitored via the Four Minute Engineering Telemetry RDR

Parameter	Comments	Units
CRC of Four Minute Engineering Telemetry RDR	Used to trigger check of tunable parameters	NA
Spectral Calibration Parameters	Used to calculate metrology laser frequency	NA
Laser Pulses Counted Per Sweep	Used to calculate metrology laser frequency	unitless
Number Of Neon Calibration Sweeps	Used to calculate metrology laser frequency	unitless
Effective Neon Wavelength	Used to calculate metrology laser frequency	nm
Neon Calibration Time Stamp	Used to calculate metrology laser frequency	ms
Repeat Neon Calibration Interval	Used to calculate metrology laser frequency	ppm
Neon Calibration Data	One sample per Number Of Neon Calibration Sweeps	NA
Starting Count (Sample 1..128)	One sample per Number Of Neon Calibration Sweeps	unitless
Starting Partial Count (Sample 1..128)	One sample per Number Of Neon Calibration Sweeps	unitless
Fringe Count (Sample 1..128)	One sample per Number Of Neon Calibration Sweeps	unitless
Ending Partial Count (Sample 1..128)	One sample per Number Of Neon Calibration Sweeps	unitless
Ending Count (Sample 1..128)	One sample per Number Of Neon Calibration Sweeps	unitless

Table 6. Tunable Parameters Provided via the Four Minute Engineering Telemetry RDR

Parameter	Comments	Units
Effective Neon Bulb Wavelength	Used in the computation of the laser frequency using Neon calibration data.	nm
Metrology laser wavelength offset	MW and SW offset from the LW metrology laser wavelength	ppm
ILS curve fit parameters	Used to correct for modulation efficiency variation with OPD	unitless
ILS FOV offset and size parameters	Use to correct off-axis pixel self apodization effects	micro radians
ICT Emissivity	Geometric factor to convert surface emissivity to effective emissivity	unitless
LW ICT Emissivity table	Wavenumber specific effective emissivities	unitless
MW ICT Emissivity table	Wavenumber specific effective emissivities	unitless
SW ICT Emissivity table	Wavenumber specific effective emissivities	unitless
Polarization Calibration	% Polarization difference relative to ICT at a specific wavenumber	unitless
Polarization Wave Numbers	Wavenumbers at which polarization calibration information is provided	cm-1
ICT environment model	Emissivities and view angles of instrument component included in the ICT radiance calculation	Temperature: K; Emissivity: unitless; View angle: degree; Planck function: mw / (m2.sr. cm-1)
ScanBaffleTemperatureBias	Scan baffle temperature correction applied at different spacecraft orbital positions (21 values)	K
Science TLM conversion coefficients	Engineering unit conversion coefficients associated with parameters monitored in eight second science/calibration telemetry RDR	unitless
Science TLM Limits	Limits associated with parameters monitored in eight	unitless

	second science/calibration telemetry RDR	
Mapping Parameters		
SSM crosstrack positions	Angles relative to SSM mounting feet	degree
SSM mirror misalignment	Mirror mount pitch and yaw errors	degree
SSM in-track position	Intrack commanded Nadir offset	degree
SSMR, SSMF, IAR , IFM boresight & SBF alignments	Alignment angles	degree
Time stamp bias	Bias added to interferogram time stamp to account for intrack motion compensation	ms
CrIS Bit Trim Mask LW	Used to reverse the bit trimming of interferogram packets	digital counts
CrIS Bit Trim Mask MW	Used to reverse the bit trimming of interferogram packets	digital counts
CrIS Bit Trim Mask SW	Used to reverse the bit trimming of interferogram packets	digital counts
LW Data Extraction Information		
Number of A/D samples	Used to reverse the bit trimming of interferogram packets	unitless
Decimation & number of filter taps	Used to reverse the bit trimming of interferogram packets	unitless
MW Data Extraction Information		
Number of A/D samples	Used to reverse the bit trimming of interferogram packets	unitless
Decimation & number of filter taps	Used to reverse the bit trimming of interferogram packets	unitless
SW Data Extraction Information		
Number of A/D samples	Used to reverse the bit trimming of interferogram packets	unitless
Decimation & number of filter taps	Used to reverse the bit trimming of interferogram packets	Unitless
Nonlinearity Correction Parameters		
linearityCorrectionA2Parameters	2nd order IR channel nonlinearity characterization parameter. There are nine linearity correction parameters for each FOV for LWIR, MWIR, and SWIR	1/Volts
linearityCorrectionVinstParameters	Detector preamp output voltage due only from instrument background radiance & detector dark current. There are nine linearity correction vinst parameters for LWIR, MWIR, and SWIR	Volts
linearityCorrectionModEffParameters	Interferometer modulation efficiency. There are nine linearity correction modeff parameters for LWIR, MWIR, and SWIR	Volts
FIR Tailoring	LWIR FIR filter coefficient gain relative to baseline FM1_3 FIR filter coefficients. Excludes gain changes due to bit trim mask changes (LW, MW and SW).	Unitless
PGA Gain Table Map	LWIR, MWIR and SWIR channel electrical gain in Volt/Volt corresponding to PGA (16x3 numbers)	Volt/Volt Config

In addition to the parameters provided through the instrument telemetry RDRs, there are many configuration options that modify the processing performed by the operational code. These values are modified via the configuration files provided by DMS. The tunable parameters provided via the configuration files are shown in Table 7.

Table 7. Tunable Parameters Provided via Configuration Files

Setting Category & Setting	Description	Units
Calibration Control Parameters		
allowCalibrationTargetDataMissing	Allows for missing ICT/DS reference measurements	boolean
allowEngineeringDataPacketsMissing	Allows for missing reference measurements	boolean
allowScienceTlmDataMissing	Allows for missing reference measurements	boolean
allowSpaceTargetTemperatureDataMissing	Allows for missing reference measurements	boolean
calibrationTargetDataValidityDuration	Absolute temporal displacement to ES under calibration	sec
calibrationTargetDataValidityDurationTolerance	Max temporal displacement of FOR under calibration	sec
disableTimeStampBasedMovingWindow	Adds additional constraints to packet timing	boolean
DsTemperatureOrigin	Specifies origin for file	CCS = 0, CONFIG = 1 CTC = 2,

elapsedTimeForValidScienceTlmData	Absolute temporal displacement to ES under calibration	sec
elapsedTimeForValidSpaceTargetTemperature	Max temporal displacement of FOR under calibration	sec
IctEmissivityOrigin	Specifies emissivities are used from config file	CONFIG = 0 TLM = 1
InstrumentTemperatureOrigin	Specifies source of value	CONFIG = 0 TLM = 1
maximumNumberOfFceTriesDuringIctDsSynchronization	Max fringe counts to try in both directions	Count
maximumNumberOfIctDsSynchronizationTries	Max ES window depth to seek valid measurement	integer
MovingAverageWindowSize	Specifies the reference window size (ES are half that)	integer
PerformRadiometricCalibration	Allows for radiometric calibration	boolean
scienceTlmTimeDifferenceTolerance	Absolute temporal displacement	sec
SkipIctDsPhaseSynchronization	Phase aligns initial ICT/DS reference windows	boolean
spaceTargetTemperatureTimeDifferenceTolerance	Max temporal displacement for temperature correlation	sec
useDeepSpaceRadiance	Specifies calibration equation to consider cold target	boolean
useIctEnvironmentalCorrectionModel	Sets ICT temp to include component contributions	boolean
useWavenumberDependentDsEmissivity	Specifies emissivities are used from config file	boolean
useWavenumberDependentIctEmissivity	Specifies emissivities are used from config file	boolean
monitorLunarIntrusion	Discards DS measurements about a threshold	boolean
maxLunarRadianceRatio	Discards DS measurements above this threshold	float
Correction Control Parameters		
applyIlsFovEffectsCorrection	Specifies the application of ILS corrections	boolean
applyIlsResidualEffectCorrection	Specifies the application of ILS residual corrections	boolean
applyPolarizationCorrections	Specifies the application of scene specific corrections	Boolean
applyPostCalibrationFilterMatrixCorrection	Specifies the application of matrix correction	boolean
ApplyResamplingMatrix	Specifies the application of resampling corrections	boolean
disableLaserMonitoring	Specifies the monitoring for laser drift	boolean
edrLwDeltaSigma	Specifies wavenumber spacing for Resampling for LW See Note	cm^{-1}
edrLwMaximumWavenumber	Specifies the high clipping range for LW See Note	cm^{-1}
edrLwMinimumWavenumber	Specifies the low clipping range for LW See Note	cm^{-1}
edrLwNumberOfPoints	Specifies the number of points in range for LW See Note	integer
edrMwDeltaSigma	Specifies wavenumber spacing for Resampling for MW See Note	cm^{-1}
edrMwMaximumWavenumber	Specifies the high clipping range for MW See Note	cm^{-1}
edrMwMinimumWavenumber	Specifies the low clipping range for MW See Note	cm^{-1}
edrMwNumberOfPoints	Specifies the number of points in range for MW See Note	integer
edrSwDeltaSigma	Specifies wavenumber spacing for Resampling for SW See Note	cm^{-1}
edrSwMaximumWavenumber	Specifies the high clipping range for SW See Note	cm^{-1}
edrSwMinimumWavenumber	Specifies the low clipping range for SW See Note	cm^{-1}

edrSwNumberOfPoints	Specifies the number of points in range for SW See Note	integer
impulseNoiseCountThreshold	Specifies limit to flag	Digital counts
laserDiodeWavelengthOrigin	Identifies the source for measurement (telemetry or config)	CONFIG = 0 TLM = 1
performFringeCountErrorHandling	Enables FCE Handling	boolean
performPolarizationCorrection	Allows for polarization correction	boolean
performSpectralAndSpatialCorrection	Allows spectral and spatial corrections	boolean
useSavedMatrices	Allows for use of saved matrices	boolean
clipGuardBands	Set to clip guard bands	boolean
NOTE: The identified parameters affect the dimensions of the CMO; the User Grid (i.e. EDR channels) must be consistent with the dimensions of the CMO at all times or the corrections will not be appropriate.		
Instrument Characteristics Parameters		
forIdentifierDs	Specifies the DS of the ICT reference measurement	integer
forIdentifierIct	Specifies the FOR of the ICT reference measurement	integer
forwardSweepDirectionLabel	“F” for Forward	char
forwardSweepDirectionIdentifier	“0” by convention	integer
laserDiodeWavelength	Laser frequency used in absence of measurement	nm
lwBandLabel	“LW”	char
lwBenchMeanIctEmissivity	LW ICT Emissivity (Instrument Location = Bench)	double
lwChamberMeanIctEmissivity	LW ICT Emissivity (Instrument Location = Chamber)	double
lwDataPointsDecimatedInterferogram	LW data points decimated	integer
lwDataPointsUndecimatedInterferogram	LW data points undecimated	integer
lwDecimationFactor	LW decimation factor	integer
meanIctEmissivity_Location	Instrument Location = Bench or Chamber	CONFIG = 0 TLM = 1
mwBandLabel	“MW”	char
mwBenchMeanIctEmissivity	MW ICT Emissivity (Instrument Location = Bench)	double
mwChamberMeanIctEmissivity	MW ICT Emissivity (Instrument Location = Chamber)	double
mwDataPointsDecimatedInterferogram	MW data points decimated	integer
mwDataPointsUndecimatedInterferogram	MW data points undecimated	integer
mwDecimationFactor	MW decimation factor	integer
numberFOR	Number of FOR	integer
numberFOV	Number of FOV	integer
numberSamplesPerLaserWavelength	Number of samples per wavelength	integer
numberSpectralBands	Number of spectral bands	integer
reverseSweepDirectionIdentifier	“1” by convention	integer
reverseSweepDirectionLabel	“R” for reverse	char
spaceTargetTemperatureDriftLimit	Specifies limit to flag	K
swBandLabel	“SW”	char
swBenchMeanIctEmissivity	SW ICT Emissivity (Instrument Location = Bench)	double
swChamberMeanIctEmissivity	SW ICT Emissivity (Instrument Location = Chamber)	double
swDataPointsDecimatedInterferogram	SW data points decimated	integer
swDataPointsUndecimatedInterferogram	SW data points undecimated	integer
swDecimationFactor	SW decimation factor	integer
engineeringCalibrationFrameType	The frame type of the CCSDS Engineering packet	unitless
scienceCalibrationFrameType	The frame type of the CCSDS Science packet	unitless
Algorithm Control Parameters		
computedWavelengthRejectionThreshold	Threshold used to reject laser wavelengths during Neon Calibration	ppm
fceParamDefaultDetectorBand	FCE default detector band	integer
fceParamDefaultDetectorFOV	FCE default detector FOV	integer

fceParamLwAmpThreshRejectLimit	Fringe Count validation parameter	double
fceParamLwDimensionThresholdLimit	Fringe Count validation parameter	double
fceParamLwFractionalFceThresholdLimit	Fringe Count validation parameter	double
fceParamLwGoodLinearFittingThreshLimit	Fringe Count validation parameter	rad^2
fceParamLwMaxFceThreshLimit	Fringe Count validation parameter	double
fceParamLwMaxIndex	Max index used in FCE detection	integer
fceParamLwMinIndex	Min index used in FCE detection	integer
fceParamMwAmpThreshRejectLimit	Fringe Count validation parameter	double
fceParamMwDimensionThresholdLimit	Fringe Count validation parameter	double
fceParamMwFractionalFceThresholdLimit	Fringe Count validation parameter	double
fceParamMwGoodLinearFittingThreshLimit	Fringe Count validation parameter	rad^2
fceParamMwMaxFceThreshLimit	Fringe Count validation parameter	double
fceParamMwMaxIndex	Max index used in FCE detection	integer
fceParamMwMinIndex	Min index used in FCE detection	integer
fceParamSwAmpThreshRejectLimit	Fringe Count validation parameter	double
fceParamSwDimensionThresholdLimit	Fringe Count validation parameter	double
fceParamSwFractionalFceThresholdLimit	Fringe Count validation parameter	double
fceParamSwGoodLinearFittingThreshLimit	Fringe Count validation parameter	rad^2
fceParamSwMaxFceThreshLimit	Fringe Count validation parameter	double
fceParamSwMaxIndex	Max index used in FCE detection	integer
fceParamSwMinIndex	Min index used in FCE detection	integer
ictPrt1Bias	Parameters use to calculate ICT Temperature	K
ictPrt2Bias	Parameters use to calculate ICT Temperature	K
laserWavelengthDriftTolerance	Used to determine if calculated laser wavelength should replace existing laser wavelength	ppm
maximumFractionRejections	Fringe Count validation parameter	double
numberOpdOverscanSamples	Number of samples to trim from each end of the interferogram	integer
polarizationCorrectionFitOrder	Order of Polynomial fit used to calculate Polarization Curve	integer
postCalibrationLwA1 postCalibrationLwA2 postCalibrationLwA3 postCalibrationLwA4	LW Parameters used to calculate Post Calibration correction matrix	integer
postCalibrationLwK0 postCalibrationLwK1 postCalibrationLwK2	LW Parameters used to calculate Post Calibration correction matrix	integer
postCalibrationMwA1 postCalibrationMwA2 postCalibrationMwA3 postCalibrationMwA4	MW Parameters used to calculate Post Calibration correction matrix	integer
postCalibrationMwK0 postCalibrationMwK1 postCalibrationMwK2	MW Parameters used to calculate Post Calibration correction matrix	integer
postCalibrationSwA1 postCalibrationSwA2 postCalibrationSwA3 postCalibrationSwA4	SW Parameters used to calculate Post Calibration correction matrix	integer
postCalibrationSwK0 postCalibrationSwK1 postCalibrationSwK2	SW Parameters used to calculate Post Calibration correction matrix	integer
timingSequenceErrorThreshold	Amount of time scan start times are allowed to vary from eight seconds with respect to adjacent scans’ start times	sec
invalidNeonCalibrationPercentageThreshold	Percentage of the number of scans by the number of EV FORs by the number of FOVs by the number of bands neon calibration values are allowed to change	double
NUM_OF_VALID_PRT_TEMP_THRESHOLD	Threshold used to indicate upper bound of quality flag numberOfValidPRTTemps[][] (cf. Table 41)	unitless
IMPULSE_NOISE_COUNT_THRESHOLD	Threshold used to indicate upper bound of	Digital counts

	quality flag ESRdrImpulseNoise[][][][] (cf. Table 41)	
ICT_TEMP_STABILITY_THRESHOLD	Threshold used to indicate lower bound of quality flag ICT_TemperatureStability[][] (cf. Table 41)	K
ICT_TEMP_CONSISTENCY_THRESHOLD	Threshold used to indicate upper bound of quality flag ICT_TemperatureConsistency[] (cf. Table 41)	K
ICT_TEMP_LOW_THRESHOLD	Lower bound for ICT temperature epoch value	K
ICT_TEMP_HIGH_THRESHOLD	Upper bound for ICT temperature epoch value	K
ShiftFactorArray	399 for FOVs 1,3,7.9 216 for FOVs 2,4,6,8 17 for FOV 5	ppm
AppShiftFactorFlag	Perform ICT spectral grid shift control parameter	boolean
performLWLinearityCorrectionControl	Perform LW linearity correction control parameter (True)	boolean
performMWLinearityCorrectionControl	Perform MW linearity correction control parameter (True)	boolean
performSWLinearityCorrectionControl	Perform SW linearity correction control parameter (False)	boolean
lwLinearityCorrectionControlParam	LW Linearity correction control parameter (TLM)	unitless
mwLinearityCorrectionControlParam	MW Linearity correction control parameter (TLM)	unitless
swLinearityCorrectionControlParam	SW Linearity correction control parameter (TLM)	unitless
surfaceEmissivityCoeff	Surface Emissivity Coefficients	Unitless
suppressSsmBaffleProfile	Suppress the SSM Baffle Profile	boolean
earthTargetTempBench	Earth Target Temperature Bench	Unitless
earthTargetTempChamber	Earth Target Temperature Chamber	Unitless
ictBaffleViewFactor	ICT Baffle View Factor	Unitless
scanBaffleViewFactor	Scan Baffle View Factor	Unitless
omaFrameViewFactor	OMA Frame View Factor	Unitless
warmBeamsplitterViewFactor	Warm Beam Splitter View Factor	Unitless
coldBeamsplitterViewFactor	Cold Beam Splitter View Factor	Unitless
earthTargetViewFactor	Earth Target View Factor	Unitless
overrideEarthTargetTemp	Overrides the SSM Target Emissivity	boolean
durationOfOrbit	Duration of the satellite orbit	
ictBaffleEmissivity	ICT Baffle Emissivity	Unitless
scanBaffleEmissivity	Scan Baffle Emissivity	Unitless
omaFrameEmissivity	OMA Frame Emissivity	Unitless
earthTargetEmissivity	Earth Target Emissivity	Unitless
scanBaffleTempOffset	Scan Baffle Temperature Offset	Kelvin
linearityCorrectionParameter_a2	Linearity Correction A2 Parameters	
linearityCorrectionVinstParameters	Linearity Correction Voltage Parameters	
linearityCorrectionParameter_ModEff	Linearity Correction Mod Effectivity Parameters	
firAccumulatorStartBit	FIR Accumulator start bits	Unitless
firFilterScaleFactor	FIR Filter Scale Factor	Unitless
firPassBandStartValues	FIR Pass Band Start Values	Unitless
firPassBandStopValues	FIR Pass Band Stop Values	Unitless
firFilterResponse_Real	FIR Filter Response Real Values	Unitless
firFilterResponse_Imag	FIR Filter Response Imaginary Values	Unitless
firEffectiveGainSetting	FIR Effective Gain Setting	Unitless
firGainMapping	FIR Gain Mapping	Unitless
laserDiodeWavelengthMW	Midwave Laser Diode Wavelength	
laserDiodeWavelengthSW	Shortwave Laser Diode Wavelength	

The CrIS Fill Packet Lookup Table contains templates of each of the Earth Scene, Deep Space, and Internal Calibration Target Interferogram packets. These templates are used to create “fill” packets that are used to replace packets missing from the CrIS RDR inputs, in order to minimize the effect of missing packets to the CrIS sliding window processing. Additional details on the CrIS Fill Packet LUT are found in the CDFCB-X Volume VIII – Look Up Table Formats.

2.1.1.1.1 Requirements for Input

The primary instrument output (interferogram data) is generated as follows:

- The instrument can perform a new measurement (sweep) every 200 ms: 167 ms for data collection and 33 ms for repositioning.
- A new cycle (scan) is repeated every 8 seconds:
  - 600 msec slew from ICT to Earth Scene (ES) 1
  - 200 msec x 30 Earth Scenes
  - 200 msec slew from ES 30 to Deep Space (DS)
  - 200 msec x 2 samples at DS (forward and reverse sweeps)
  - 400 msec slew from DS to ICT
  - 200 msec x 2 samples at ICT (forward and reverse sweeps)
- Each scan is comprised of 918 interferograms: (2 DS + 2 ICT + 30 ES) × 3 bands × 9 detectors / band

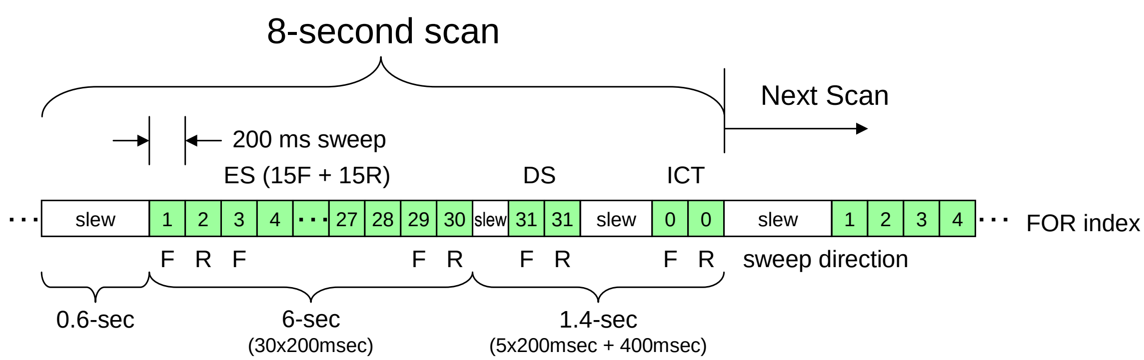

Figure 3. CrIS Measurement Sequence

2.1.1.1.2 Requirements for Applicable Auxiliary/Ancillary and/or Optional Input Data

In addition to the interferogram data generated by the CrIS Sensor, there are two additional packets of interest to the SDR algorithm:

1. Eight second Science/Calibration Telemetry – This data packet contains instrument temperature information required during the calibration process collected during the previous eight seconds of instrument operation. These data packets are transmitted by the instrument at the end of every eight second scan period.
2. Four Minute Engineering Telemetry – This data packet contains additional information used for interferogram processing, spectral calibration, spatial correction and geolocation. With the exception of the Neon calibration data contained within this packet, all information is virtually static; however it can be modified via table upload commands sent to the CrIS sensor. Neon calibration data is updated as programmed and is currently planned for once per orbit. These data packets are transmitted by the instrument at the end of every thirtieth eight second scan period.

2.1.1.2 Outputs

The CrIS SDR algorithm creates two outputs and stores those in DMS. One output contains Geolocation information and the other output contains unapodized radiance information. Table

8 shows the CrIS SDR outputs. Refer to the CDFCB-X, D34862, for a detailed description of the outputs.

Table 8. CrIS SDR Outputs

Output	Description
CrIS SDR Data	This contains the unapodized CrIS radiances (See Table 9)
CrIS Geolocation Data	Contains the geolocation information (See Table 10)
CrIS SDR DQN	Data quality notifications for SDR and RDR yields (percentage of good pixels)
CrIS Terrain-Correction Geolocation	This product is the same format as Table 10 except there are two fields for the roll and pitch angles at the beginning of the structure. Each of these has dimensions of 4*30*9 (corresponding to scans, earth scenes, and fields of view, respectively) and is a 4-byte floating point field.
Correction matrix data / Engineering calibration record	Correction matrix last calculated to speed up processing and also the last received engineering calibration record. This item will only be output if the input item is missing or if a new version is created by the algorithm. Therefore, this item is a possible input and/or output of the algorithm.

The current CrIMSS EDR Algorithm expects to read in the following input information from the CrIS SDRs:

CrIMSS EDR Algorithm Input from CrIS SDR Algorithm Output	
FOV Location	Longitude (degrees) Latitude (degrees)
Sensor Scan Angle	Sensor Scan Angle [degrees]
Radiance	Radiance value (mW/m2/Sr/cm-1)

The CrIS SDR algorithm shall produce unapodized radiances to be compliant with CrIS SDR product requirements. The CrIMSS EDR algorithm is responsible for applying the Blackman apodization. The only problem is that information is lost at the two ends of each of the three CrIS bands due to convolution. This can be avoided, however, by extending the spectral grid of the unapodized radiances.

<edrLwDeltaSigma Default_”0.625000”/>
<edrLwMaximumWavenumber Default=”**1096.250000**”/>
<edrLwMinimumWavenumber Default=”**648.750000**”/>
<edrLwNumberOfPoints Default=”**717**”/>
<edrMwDeltaSigma Default=”1.250000”/>
<edrMwMaximumWavenumber Default=”**1752.500000**”/>
<edrMwMinimumWavenumber Default=”**1207.500000**”/>
<edrMwNumberOfPoints Default=”**437**”/>
<edrSwDeltaSigma Default=”2.500000”/>
<edrSwMaximumWavenumber Default=”**2555.000000**”/>
<edrSwminimumWavenumber Default=”**2150.000000**”/>
<edrSwNumberOfPoints Default=”**163**”/>

As a result of the change, the CrIS SDR algorithm outputs unapodized radiances for the following spectral grid:

LWIR Band: a total of 717 bins from 648.75 cm-1 to 1096.25 cm-1 with a spacing of 0.625 cm-1;
MWIR Band: a total of 437 bins from 1207.5 cm-1 to 1752.5 cm-1 with a spacing of 1.25 cm-1;
SWIR Band: a total of 163 bins from 2150 cm-1 to 2555 cm-1 with a spacing of 2.5 cm-1.

Comparing to the original CrIS SDR spectral grid, the new one added two more grid points to the ends of each of the three CrIS bands. With the 12 newly added bins, the total number of bins of the CrIS SDR output radiances has increased from 1305 to 1317. The apodized radiances for the three CrIS bands are concatenated into one spectrum with a total of 1305 bins in the CrIMSS EDR algorithm.

Table 9. CrIS SDR Output Structure

Name	Type and Dimensions	Description	Units / Range / Fill
ESRealLW	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [LWNUMBEROFPOINTS]	CrIS LW Band spectrally and radiometrically calibrated radiances (real part of spectra)	mW/(m^2 sr cm^{-1}) / 1.17549e-38 to 3.40282e+38 / -999.4
ESRealMW	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [MWNUMBEROFPOINTS]	CrIS MW Band spectrally and radiometrically calibrated data for mid-wave band (real part of spectra)	mW/(m^2 sr cm^{-1}) / 1.17549e-38 to 3.40282e+38 / -999.4
ESRealSW	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [SWNUMBEROFPOINTS]	CrIS SW Band spectrally and radiometrically calibrated data for short-wave band (real part of spectra)	mW/(m^2 sr cm^{-1}) / 1.17549e-38 to 3.40282e+38 / -999.4
ESImaginaryLW	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [LWNUMBEROFPOINTS]	Imaginary part of spectra for long-wave band	mW/(m^2 sr cm^{-1}) / 1.17549e-38 to 3.40282e+38 / -999.4
ESImaginaryMW	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [MWNUMBEROFPOINTS]	Imaginary part of spectra for mid-wave band	mW/(m^2 sr cm^{-1}) / 1.17549e-38 to 3.40282e+38 / -999.4
ESImaginarySW	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [SWNUMBEROFPOINTS]	Imaginary part of spectra for short-wave band	mW/(m^2 sr cm^{-1}) / 1.17549e-38 to 3.40282e+38 / -999.4
ESNEdNLW	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [LWNUMBEROFPOINTS]	Spectral Noise Estimate - long-wave	mW/(m^2 sr cm^{-1}) / 1.17549e-38 to 3.40282e+38 / -999.4
ESNEdNMW	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [MWNUMBEROFPOINTS]	Spectral Noise Estimate - mid-wave	mW/(m^2 sr cm^{-1}) / 1.17549e-38 to 3.40282e+38 / -999.4
ESNEdNSW	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV]	Spectral Noise Estimate - short-wave	mW/(m^2 sr cm^{-1}) / 1.17549e-38 to 3.40282e+38 / -999.4

Name	Type and Dimensions	Description	Units / Range / Fill
	[SWNUMBEROFPOINTS]		
DS_WindowSize	UInt16 [CRIS_SCAN_PER_GRAN] [BOTHDIRECTIONS] [CRIS_MAX_FOV] [CRIS_TOTAL_BANDS]	The number of Deep Space (DS) spectra used to calibrate the earth scene	Unitless / 0 to 65529 / 65530
ICT_WindowSize	UInt16 [CRIS_SCAN_PER_GRAN] [BOTHDIRECTIONS] [CRIS_MAX_FOV] [CRIS_TOTAL_BANDS]	The number of Internal Calibration Target (ICT) spectra used to calibrate the earth scene	Unitless / 0 to 65529 / 65530
ESZPDMagnitude	UInt16 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [CRIS_TOTAL_BANDS]	Interferogram magnitude at zero path difference	Unitless / 0 to 65529 / 65530
ESZPDFringeCount	UInt16 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [CRIS_TOTAL_BANDS]	Interferogram fringe count at zero path difference before decimation	Unitless / 0 to 65529 / 65530
sdrFringeCount	UInt16 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [CRIS_TOTAL_BANDS]	The calculated number of fringes that the interferogram was advanced or delayed	Unitless / 0 to 65529 / 65530
ESRdrImpulseNoise	UInt8 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [CRIS_TOTAL_BANDS]	This flag represents the number of samples in an interferogram that exceeded the impulse noise mask and were set to zero; if > 1 the resultant spectrum is flagged as having excess noise	Unitless / 0 to 249 / 250
monitoredLaserWavelength	Float64 [CRIS_SCAN_PER_GRAN]	This flag represents the monitored laser metrology wavelength, calculated using data from the 4-min engineering packets and Neon calibrated laser metrology wavelength	nm / 2.22507e-308 to 1.79769e+308 / -999.4
measuredLaserWavelength	Float64 [CRIS_SCAN_PER_GRAN]	This quality flag represents the measured metrology laser wavelength with neon lamp calibration	nm / 2.22507e-308 to 1.79769e+308 / -999.4
resamplingLaserWavelength	Float64 [CRIS_SCAN_PER_GRAN]	This flag represents the wavelength used for the spectral resampling, which is half of the current metrology laser wavelength	nm / 2.22507e-308 to 1.79769e+308 / -999.4
DS_Symmetry	Float64 [CRIS_SCAN_PER_GRAN] [CRIS_MAX_FOV]	This flag is intended to identify the asymmetry in the	Unitless / 2.22507e-308 to 1.79769e+308 / -999.4

Name	Type and Dimensions	Description	Units / Range / Fill
	[CRIS_TOTAL_BANDS]	measured DS IGMs	
DS_SpectralStability	Float64 [CRIS_SCAN_PER_GRAN] [DS_FOR_PER_SCAN] [CRIS_MAX_FOV] [CRIS_TOTAL_BANDS]	This flag monitors the spectral variability of the DS views within the moving window	Unitless / 2.22507e-308 to 1.79769e+308 / -999.4
ICT_SpectraStability	Float64 [CRIS_SCAN_PER_GRAN] [IC_FOR_PER_SCAN] [CRIS_MAX_FOV] [CRIS_TOTAL_BANDS]	This flag monitors the spectral variability of the ICT views within the moving window	Unitless / 2.22507e-308 to 1.79769e+308 / -999.4
ICT_TemperatureStability	Float32 [CRIS_SCAN_PER_GRAN] [IC_FOR_PER_SCAN]	This flag measures the stability of the two Platinum Resistance Temperature measurements of the Internal Calibration Target	Unitless / 1.17549e-38 to 3.40282e+38 / -999.4
ICT_TemperatureConsistency	Float32 [CRIS_SCAN_PER_GRAN]	This flag measures the consistency between the two Platinum Resistance Temperature measurements of the Internal Calibration Target	Unitless / 1.17549e-38 to 3.40282e+38 / -999.4
numberOfValidPRTTemps	UInt8 [CRIS_SCAN_PER_GRAN] [BOTHDIRECTIONS]	Number of valid PRT Temperatures used	Unitless / 0 to 249 / 250
QF1	UInt8 [CRIS_SCAN_PER_GRAN]	Scan-level Quality Flags	Unitless / See Table 28 / None
QF2	UInt8 [CRIS_SCAN_PER_GRAN] [CRIS_MAX_FOV] [CRIS_TOTAL_BANDS]	FOV-level Quality Flags	Unitless / See Table 27 / None
QF3	UInt8 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [CRIS_TOTAL_BANDS]	Band-level Quality Flags (Band 1)	Unitless / See Table 25 / None
QF4	UInt8 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV] [CRIS_TOTAL_BANDS]	Band-level Quality Flags (Band 2)	Unitless / See Table 26 / None

Table 10. CrIS SDR Geolocation Output Structure

Name	Type and Dimensions	Description	Units / Range / Fill
FOR_IETusec	Int64 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN]	Individual IET times for each FOR	Microseconds / 0 to 5680281632999999 / -998
scanStartTime	Int64 [CRIS_SCAN_PER_GRAN]	Starting Time of Scan in IET	Microseconds 0 to 5680281632999999 / -998
scanMidTime	Int64 [CRIS_SCAN_PER_GRAN]	Mid Time of Scan in IET	Microseconds 0 to 5680281632999999 / -998
lat	Float32	Latitude - positive north.	Degrees /

Name	Type and Dimensions	Description	Units / Range / Fill
	[CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV]		-90 to 90 / -999.8
lon	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV]	Longitude - Positive east.	Degrees / -180 to 180 / -999.8
sunzen	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV]	Solar Zenith Angle at the geolocated FOV center.	Degrees / 0 to 180 / -999.8
sunazm	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV]	Solar Azimuth Angle at the geolocated FOV center.	Degrees / -180 to 180 / -999.8
satzen	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV]	Satellite Zenith Angle at the geolocated FOV center	Degrees / 0 to 180 / -999.8
satazm	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV]	SatelliteAzimuth Angle at the geolocated FOV center. Positive east of north.	Degrees / -180 to 180 / -999.8
height	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV]	Ellipsoid-Geoid separation	Meters / 800,000 to 2,000,000 / -999.8
range	Float32 [CRIS_SCAN_PER_GRAN] [EV_FOR_PER_SCAN] [CRIS_MAX_FOV]	Line of sight distance from the ellipsoid intersection to the satellite	Meters / 800,000 to 2,000,000 / -999.8
scPosition	Float32 [CRIS_SCAN_PER_GRAN] [VEC_SIZE]	Spacecraft position in ECR Coordinates (X, Y, Z) at the midtime of scan.	Meters / -7460000 to 7400000 / -999.8
scVelocity	Float32 [CRIS_SCAN_PER_GRAN] [VEC_SIZE]	Spacecraft velocity in ECR Coordinates (dx/dt, dy/dt, dz/dt) at the midtime of scan	Meters/Second / -7500 to 7500 / -999.8
scAttitude	Float32 [CRIS_SCAN_PER_GRAN] [VEC_RPY_SIZE]	The relative orientation of the Spacecraft with respect to the Geodetic Reference Frame at the midtime of scan	Arcseconds / -300 to 300 / -999.8
geoQualFlag	UInt8 [CRIS_SCAN_PER_GRAN]	Attitude and Ephemeris Availability Status	Unitless / See Table 11 / None

Table 11. CrIS SDR Geolocation Output Structure geoQualFlag Details

Bits	Description	Legend Entries
0-1	Attitude and Ephemeris Availability Status	0 = Nominal – E&A data available 1 = Missing Data <= Small Gap 2 = Small Gap < Missing Data <= Granule Boundary 3 = Missing Data > Granule Boundary
2-7	Spare	N/A

2.1.2 Algorithm Processing

CrIS SDR processing gets the ingest Raw Data Records (RDRs) from DMS. CrIS RDR data in DMS is in CCSDS packet format. The CrIS SDR algorithm is tasked to process a granule of RDR data. In order to do that, it needs to retrieve 15 scans prior to the beginning of the granule and 15 scans after the end of the granule. The first step is to determine the beginning scan in the current granule and then determine the granule IDs for the RDR data on both sides of the granule. Once the RDR files have been retrieved and the time stamp for the beginning scan determined, a structure of packet pointers is populated with pointers to the actual packets in the RDR files based on scans and time. This structure of packet pointers is passed into the CrIS SDR algorithm which parses and processes the packets. Common geolocation routines are used to get the location by providing the exit vector for that sample so that geolocation information can be updated. For information regarding common geolocation refer to D41869 (current revision)_VIIRS_CmnGeo_OAD.doc. The specific common geolocation routines used, here, are satPosAtt() to locate the ephemeris and attitude for the specific observation time, and ellipIntersect() to calculate the geodetic latitude and longitude and terrain height for the given sensor exit vector and ephemeris and attitude. The exit vectors are calculated with

$$E_x = -\sin(P_{LOS})$$
$$E_y = \cos(P_{LOS}) * \sin(R_{LOS})$$
$$E_z = \cos(P_{LOS}) * \cos(R_{LOS})$$

Where P_{LOS} and R_{LOS} are the line-of-sight roll and pitch and $\vec{E}$ is the exit vector.

2.1.2.1 Main Module - SDR Generator Application

This paragraph outlines the major software contributors used to process NPOESS CrIS data. Detailed design relationships of these classes can be obtained from the design model. Detailed implementation details are documented in the source with in-line comments. Table 12 lists some important classes used during processing.

Table 12. Some Important Classes Used During Processing

Name	Type	Initialization	Description
CalibratedSpectra	Class	Instantiation	Specialization of Spectra which implements the complex calibration interface
CCSDSFormat	Class	Singleton	Utility for packaging and extraction of CCSDS primary and secondary headers
CorrectionMatrix	Class	Instantiation	Container for the spatial corrections that comprise the CMO
EngineeringCalibrationRecord	Class	Specialization	Implements the abstract CalibrationRecord for Algorithm specific 4min values
Interferogram	Class	Instantiation	Container for organizing video data
ScienceCalibrationRecord	Class	Specialization	Implements the abstract CalibrationRecord for Algorithm specific 8sec values
ScienceDataProcessor	Class	Instantiation	Central engine for Algorithm behavior
Spectra	Class	Abstract	Interface container
SpectraManager	Class	Instantiation	Collection of Spectra used in window average
TelemetryProcessor	Class	Singleton	Extracts binary instrument data defined by config file
TemperatureHistory	Class	Instantiation	Collection responsible for min, max, avg temperature
VideoData	Class	Aggregation	Implements TelemetryProcessor's bit-trim behavior

2.1.2.2 Class CalibratedSpectra

This class provides complex calibration and applies corrections to the instance spectra.

Table 13 shows the CalibratedSpectra attributes and Table 14 shows the CalibratedSpectra operations.

2.1.2.2.1 CalibratedSpectra Attributes

Table 13. CalibratedSpectra Attributes

Name	Type	Description
VERSION_NUMBER	primitive unsigned short int UInt16	*Collection of version number.*

2.1.2.2.2 CalibratedSpectra Operations

Table 14. CalibratedSpectra Operations

Name	Return	Parameters	
Applies a complex calibration to the instance spectra. The cold contribution is added only when coldTargetTemp is != NULL.			
calibrate	primitive bool	Class Spectra*	*hotReference*
		Class Spectra*	*coldReference*
		Class Spectra*	*hotTargetTemp*
		Class Spectra*	*coldTargetTemp*
Applies a bin by bin correction.			
correct	primitive bool	Class Spectra*	*correctionSpectra*
Return linear phase shift when fringe count errors exist.			
correctFringeCountError	*Class BOOST::vector<std::complex<Float64> >&*	Class BOOST::vector<std::complex<Float64> >&	*linearPhaseShift*
		typedef double Float64	*fringeCount*
Fringe count errors are computed and returned.			
detectFringeCountError	typedef double Float64	Class Spectra*	*hotReference*
		Class Spectra*	*coldReference*
Shifts the imaginary component of phase to align the real component with the reference spectra.			
adjustPhase	primitive void	Class BOOST::vector<std::complex<Float64> >&	*linearPhaseShift*

2.1.2.3 CCSDSFormat

This class implements the CommonMessageFormat for CCSDS packetization. Table 15 shows the CCSDSFormat attributes and Table 16 shows the CCSDSFormat operations

2.1.2.3.1 CCSDSFormat Attributes

Table 15. CCSDSFormat Operations

Name	Type	Description
thePrimaryHeader	Class CcsdsPrimaryHeader	*Collection of routing a packet and its data.*
theSecondaryHeader	Class CcsdsSecondaryHeader	*Collection of routing the timestamp portion of a CCSDS header*

2.1.2.3.2 CCSDSFormat Operations

Table 16. CCSDSFormat Operations

Name	Return	Parameters	
Populates the primary and secondary headers based on the data at the offset of startingLocation.			
unpack	typedef int Int32	Class Octets	*&theData*

Name	Return	Parameters	
		typedef unsigned int UInt32	*&startingLocation*

2.1.2.4 Class CorrectionMatrix

This class is responsible for implementing a Correction Matrix that is applied to each spectra. Table 17 shows the CorrectionMatrix attributes and Table 18 shows the CorrectionMatrix operations.

2.1.2.4.1 CorrectionMatrix Attributes

Table 17. CorrectionMatrix Attributes

Name	Type	Description
Size	typedef unsigned int UInt32	*Size of Correction Matrix*
validCorrection	primitive bool	*Valid correction flag*
childTable[ ]	Class CorrectionTable*	*Collection of child tables*
calibrationMatrix	Class BOOST::matrix<Float32>	*Collection of calibration matrix*
lastUpdateTime	typedef struct SYSTEMTIME	*Collection of last update time*
AlgorithmParameter	Class SDR_AlgorithmnParameter	*Collection of SDR algorithm parameters*
CorrectionParameter	Class SDR_CorrectionParam	*Collection of SDR correction parameters*
InstrumentCharacteristic	Class CrIS_InstrumentCharacteristics	*Collection of CrIS instrument parameters*
maxPathDifferential[ ]	typedef double Float64	*Collection of maximum path difference*
instrumentDeltaSigma[ ]	typedef double Float64	*Collection of instrument delta sigma*
instrumentLowestWavenumber[ ]	typedef double Float64	*Collection of instrument lowest wavenumber*
userDeltaSigma[ ]	typedef double Float64	*Collection of user delta sigma*
userLowestWavenumber[ ]	typedef double Float64	*Collection of user lowest wavenumber*
boostCalMatrix	typedef boost::numeric::ublas::matrix<Float64,boost::numeric::ublas::column_major> ColMatrix	*Column matrix*
boostChildTable	typedef boost::numeric::ublas::matrix<Float64> RowMatrix	*Row matrix*

2.1.2.4.2 CorrectionMatrix Operations

Table 18. CorrectionMatrix Operations

Name	Return	Parameters	
Rebuilds child tables for calibration matrix construction.			
buildTable	typedef int Int32	Class EngineeringCalibrationRecord&	*theEngCalRec*
		enum SceneElement	*theFov*
		enum Band::Wavelength	*theBand*
Builds self-apodization table.			
buildSelfApodizationTable	primitive bool	Class EngineeringCalibrationRecord&	*theEngCalRec*
		enum SceneElement	*theFov*
		enum Band::Wavelength	*theBand*
Builds Residual ILS Table.			
buildResidual_ILS_Table	primitive bool	Class EngineeringCalibrationRecord&	*theEngCalRec*
		enum SceneElement	*theFov*

Name	Return	Parameters	
		enum Band::Wavelength	*theBand*
Calculates the post calibration table. This calculation is only band specific.			
buildPostCalibrationTable	primitive bool	enum Band::Wavelength	*theBand*
Calculates Resampling Table. This calculation is only band specific.			
buildResamplingTable	primitive bool	enum Band::Wavelength	*theBand*
Calculates the user apodization table. This calculation is only band specific.			
buildUserApodizationTable	primitive bool	enum Band::Wavelength	*theBand*
Resizes the correction and child tables to new dimension.			
setSize	primitive void	typedef unsigned int UInt32	*newDimension*
Mathematical inclusion in current correction values.			
addChildContribution	primitive void	enum TableTypes	t*ableType*
Serializes the Child Tables of the CorrectionMatrix.			
serializeChildTables	primitive void	Class boost::archive::binary_oarchive	*&ar*
Serializes the Child Tables of the CorrectionMatrix.			
serializeChildTables	primitive void	Class boost::archive::binary_iarchive	*&ar*
Creates a comma separated string containing values from the calibrationMatrix.			
toString	primitive string	n/a	*n/a*
Serializes the correction matrix with user deltaSigma.			
serializeCMO	primitive void	Class boost::archive::binary_oarchive&	*ar*
		enum Band::Wavelength	*theBand*
Serializes the correction matrix with user deltaSigma.			
serializeCMO	primitive void	Class boost::archive::binary_iarchive&	*ar*
		enum Band::Wavelength	*theBand*

2.1.2.5 Class EngineeringCalibrationRecord

This class is a parent class to each of the Engineering Calibration Record sub categories. Table 19 shows the EngineeringCalibrationRecord attributes and Table 20 shows the EngineeringCalibrationRecord operations.

2.1.2.5.1 EngineeringCalibrationRecord Attributes

Table 19. EngineeringCalibrationRecord Attributes

Name	Type	Description
ietTime	typedef long long Int64	*IET time*
computedWavelengthRejectionThreshold	typedef unsigned int UInt32	*Computed wavelength rejection threshold*
rejectedWavelengthsRatio	typedef double Float64	*Rejected wavelength ratio*
averageMetrologyWavelength	typedef double Float64	*Averaged metrology wavelength*
engLaserDiodeTemperature	typedef double Float64	*Laser diode temperature*
engLaserDiodeCurrent	typedef double Float64	*Laser diode current*
laserDiodeWavelengthOrigin	enum LaserWavelengthSource	*Original laser diode wavelength*
ilsParams	typeded struct ILS_Parameters	*Collection of ILS parameters*
ilsOrigin	enum IlsOriginSource	*Collection of ILS original source*
theEngCalRec_ICTEmissivityParameters	Class EngCalRec_ICTEmissivityParameters	*Collection of engineering ICT emissivity parameters*

Name	Type	Description
theEngCalRec_ILSCurveFitParameters	Class EngCalRec_ILSCurveFitParameters	*Collection of ILS curve fit parameters*
theEngCalRec_ILSFOVParameters	Class EngCalRec_ILSFOVParameters	*Collection of ILS fov parameters*
theEngCalRec_ICTEnvironmentalModel	Class EngCalRec_ICTEnvironmentalModel	*Collection of ICT environmental model*
theEngCalRec_ScienceTelemetryConversionCoefficients	Class EngCalRec_ScienceTelemetryConversionCoefficients	*Collection of science telemetry conversion coefficients*
theEngCalRec_PolarizationCalibrationWavenumbers	Class EngCalRec_PolarizationCalibrationWavenumbers	*Collection of polarization calibration wavenumbers*
theEngCalRec_LaserMetrologyInfo	Class EngCalRec_LaserMetrologyInfo	*Collection of laser metrology information*
theEngCalRec_NeonCalInfo;	Class EngCalRec_NeonCalInfo	*Collection of neon calibration information*
theEngCalRec_ScienceTelemetryLimits	Class EngCalRec_ScienceTelemetryLimits	*Collection of science telemetry limits*
theEngCalRec_PitchRollYawInfo	Class EngCalRec_PitchRollYawInfo	*Collection of pitch, roll and yaw information*
theEngCalRec_LinearityErrorParameters	Class EngCalRec_LinearityErrorParameters	*Collection of linearity error parameters*
losInSSMF[ ]	Class BOOST::vector<Float64>	*Collection of pitch and yaw misalignments between interferometer and scan mechanism*
normalRMF	Class BOOST::vector<Float64>	*Collection of mirror normal to SSM mirror in RMF coordinate system*
sbfToSSMFMatrix	Class BOOST::vector<Float64>	*Collection of transformation from SBF to SSMF*
lastCheckSumValue	typedef int Int32	*Last check sum value*
packetVersionNumber	typedef int Int32	*Packet version number*

2.1.2.5.2 EngineeringCalibrationRecord Operations

Table 20. EngineeringCalibrationRecord Operations

Name	Return	Parameters	
Refreshes data.			
refreshData	primitive bool	typedef unsigned short int UInt16	*theApid*
		Class CCSDSFormat*	*theTelemetryFormat*
Calculates metrology wavelength.			
calculateMetrologyWavelength	primitive void	n/a	*n/a*
Loads the values from the engineering calibration record.			
serialize	primitive void	Class boost::archive::binary_oarchive	*&ar*
		Class std::ostringstream stringstream	*&oss*
Stores the values to the engineering calibration record.			
serialize	primitive void	Class boost::archive::binary_iarchive	*&ar*
		Class std::istringstream stringstream	*&oss*
Initializes the scene selection module frame (SSMF) to spacecraft body frame (SBF) transformation operator.			
initializeGeometricCalibration	primitive void	n/a	*n/a*
Computes rotation matrix around Y axis.			

Name	Return	Parameters	
rotationMatrixY	Class BOOST::matrix<Float64>	typedef double Float64	*thePitch*
Compute rotation matrix around X axis.			
rotationMatrixX	Class BOOST::matrix<Float64>	typedef double Float64	*theRoll*
Computes rotation matrix around Z axis.			
rotationMatrixZ	Class BOOST::matrix<Float64>	typedef double Float64	*theYaw*

2.1.2.6 Class Interferogram

This class is responsible for extracting interferogram data from the telemetry processor. Each interferogram consists of a vector of real parts and a vector of imaginary parts. Table 21 shows the Interferogram attributes and Table 22 shows the Interferogram operations

2.1.2.6.1 Interferogram Attributes

Table 21. Interferogram Attributes

Name	Type	Description
realSample	Class BOOST::vector<Float64>*	*Collection of real parts*
imaginarySample	Class BOOST::vector<Float64>*	*Collection of imaginary parts*
RDR_Status	typedef struct RawDataRecordStatusRegister	*Collection of raw data status*
status	typedef unsigned int UInt32	*Status*
channel	typedef unsigned int UInt32	*Channel number*
theScene	enum FieldOfRegard	*Actual field of regard reported in interferogram*
porchSwingDirection	enum SweepDirection	*Actual porch swing direction reported in interferogram*
spectralBand	enum Band::Wavelength	*Actual band reported in interferogram*
fieldOfView	enum SceneElement	*Actual field of view reported in interferogram*

2.1.2.6.2 Interferogram Operations

Table 22. Interferogram Operations

Name	Return	Parameters	
Adds an interferogram to the current interferogram.			
operator+	primitive void	Class Interferogram&	*rightOperand*
subtracts an interferogram from the current interferogram.			
operator-	primitive void	Class Interferogram&	*rightOperand*
Divides each element of this interferogram by a specified number.			
operator/	primitive void	typedef double Float64	*factor*
Copies all data from source interferogram.			
operator=	Class Interferogram&	Class Interferogram&	*source*
Outputs information of the current interferogram.			
operator<<	Class ostream&	Class ostream&	*outStream*
		Class Interferogram&	*i*
returns data of the specified type from the interferogram.			
getSamples	Class	enum Complex	*Real or imaginary samples*

Name	Return	Parameters	
	BOOST::vector<Float64>&		
Computes deep space symmetry.			
computeDeepSpaceSymmetry	typedef double Float64	Class Interferogram&	*forwardDS*
		Class Interferogram&	*reverseDS*
References to the root of the complex squares.			
getMagnitude	Class BOOST::vector <Float64>&	n/a	*n/a*

2.1.2.7 Class TemperatureHistory

This class is a container for temperatures gathered from the data. Table 23 shows the temperatureHistory attributes and Table 24 shows the temperatureHistory operations.

2.1.2.7.1 TemperatureHistory Attributes

Table 23. TemperatureHistory Attributes

Name	Type	Description
windowSize_	typedef int Int32	*Moving average window size*
historicalWindow_	Class vector<CalRecord*>	*A vector that contains average temperatures*
minTemperature_	typedef double Float64	*Minimum temperature*
maxTemperature	typedef double Float64	*Maximum temperature*

2.1.2.7.2 TemperatureHistory Operations

Table 24. TemperatureHistory Operations

Name	Return	Parameters	
Removes all records in the temperature history window.			
clearHistory	primitive void	n/a	*n/a*
Sets the window size for a temperature window.			
setWindowSize	primitive void	typedef int Int32	*newSize*
Returns a temperature at any given position.			
getTemperature	typedef double Float64	typedef int Int32	*position*
Adds the current average temperature to the window if a temperature is not supplied.			
addTemperature	typedef double Float64	n/a	*n/a*
Adds temperature to a temperature window.			
addTemperature	typedef double Float64	typedef double Float64	*newTemp*
		typedef struct CrisSdrHeapDataType	**algDataPtr*
		typedef unsigned int UInt32	*scanIdx*
		typedef int Int32	*ictTempType*
Calculates an average of a temperature window.			
calculateAverageTemperature	typedef double Float64	n/a	*n/a*
returns the number of valid temperatures in temp history window			

Name	Return	Parameters	
getValidCount	typedef int Int32	n/a	*n/a*
Removes a temperature record from the temperature window.			
removeTemperature	primitive bool	n/a	*n/a*

2.1.2.8 Class ScienceDataProcessor

This class is responsible for processing the CrIS SDR science data. Table 25 shows the ScienceDataProcessor attributes and Table 26 shows the ScienceDataprocessor operations.

2.1.2.8.1 ScienceDataProcessor Attributes

Table 25 shows the science data processor attributes. This class instantiates the Planck radiation constants for the Planck Function specified in the CrIS ATBD (D43773). They are of type double, as opposed to the single precision format specified in the ATBD. C_1 requires at least five extra points of precision, because the ATBD multiplies it by a factor of 10-5. This extra amount of precision also ensures better accuracy in Planck function calculations.

Table 25. ScienceDataProcessor Attributes

Name	Type	Description
correctionTable[ ][ ]	Class CorrectionMatrix	Collection of correction matrix tables
engPacketCount	typedef int Int32	Engineering packet count
*DMS_Data_Inputs	typedef struct CrisSdrAlgDataType	Collection of DMS input parameters
windowSize	typedef int Int32	Window size
processingPeriod	Class SpectraManager	Collection of spectra currently used in the moving window average.
polarizationCorrectionCurve	Class Spectra*	Collection of polarization correction curves
updateTargetRadiance	primitive bool	Updates target radiance
CalibrationParameter	typedef struct CalibrationParam	Collection of calibration parameters
CorrectionParameter	typedef struct SDR_CorrectionParam	Collection of SDR correction parameters
InstrumentCharacteristic	Class CrIS_InstrumentCharacteristics	Collection of CrIS instrument characteristics
TestEnvironmentParameter	typedef struct CrIS_TestEnvironmentParameters	Collection of CrIS environment parameters
GeneralParameter	typedef struct SDR_GeneralParameters	Collection of CrIS general parameters
AlgorithmParameter	struct SDR_AlgorithmnParameter	Collection of SDR algorithm parameters
maxPathDifferential[ ]	typedef double Float64	Max path differential
deltaSigma[ ]	typedef double Float64	Delta sigma
lowestWavenumber[ ]	typedef double Float64	Lowest wavenumber
validCalTargetDuration	Class TimeRange	Collection of valid calibration target time
validCalTargetTolerence	typedef struct SYSTEMTIME	Collection of valid calibration time tolerance
validColdTargetDuration	Class TimeRange	Collection of valid cold target time
validColdTargetTolerence	typedef struct SYSTEMTIME	Collection of valid cold target tolerance
validHotTargetDuration	Class TimeRange	Collection of valid hot target time
validHotTargetTolerence	typedef struct SYSTEMTIME	Collection of valid hot target tolerance
sequenceCount[ ][ ][ ][ ]	typedef int Int32	Collection of spectra sequence counts
scienceCount[ ][ ][ ][ ]	typedef int Int32	Collection of spectra science counts
targetCount[ ][ ][ ][ ]	typedef int Int32	Collection of spectra target counts

invalidInterferogramCount[ ][ ]	typedef int Int32	Collection of invalid interferogram counts
invalidSpectraCount[ ][ ]	typedef int Int32	Collection of invalid spectra counts
currentExternalCalFileLine	primitive string	Current external cal file line
theEngineeringCalibrationRecord	Class EngineeringCalibrationRecord	Collection of engineering calibration record
theTargetTempCalibrationRecord	Class TargetTempCalibrationRecord	Collection of target temperature calibration record
theScienceCalibrationRecord	Class ScienceCalibrationRecord	Collection of science calibration record
plankConstC1	typedef double Float64	Planck function constant value
plankConstC2	typedef double Float64	Planck function constant value
appDirectory	primitive string	Directory name
calculateDesiredBandCenter	primitive void	Calculate desired band center
SCIENCE_CALIBRATION_FRAME_TYPE	typedef unsigned int UInt32	Science calibration frame type
ENGINEERING_CALIBRATION_FRAME_TYPE	typedef unsigned int UInt32	Engineering calibration frame type
requiresPhaseSync[ ][ ][ ]	primitive bool	Collection of required phase synchronization
syncAttemptCount[ ][ ][ ]	typedef int Int32	Collection of attempt synchronization count
totalActiveThreads	typedef int Int32	Total active threads
threadFOVmap[ ]	typedef int Int32	Thread FOV
earthScenefceDetectionComplete[ ]	primitive bool	Collection of earth scene FCE detection completion
referenceSceneReceived[ ][ ]	primitive bool	Collection of reference scene received
referenceSceneSync[ ][ ][ ]	primitive bool	Collection of reference scene synchronization
initialCorrectionReady	primitive bool	Initial correction ready flag
FIR_GAIN[ ]	Class UncalibratedSpectra	Collection of FIR gain
scanIdx_	typedef unsigned int UInt32	Scan index
numCMOBuilds_	typedef unsigned short UInt16	Number of CMO builds
taskedSequentially_	primitive bool	Task sequential flag
scanWhichCausedRebuild_	typedef signed short Int16	Scan number that causes to rebuild CMO
currentDSfwd	Class Interferogram	Current DS forward interferogram
currentDSrev	Class Interferogram	Current DS reverse interferogram
currentDSfwd_p	primitive bool	Current DS forward flag
currentDSrev_p	primitive bool	Current DS reverse flag
forDS_reals[ ][ ][ ]	Class BOOST::vector < Float64>	Collection of forward DS real parts
forDS_imags[ ][ ][ ]	Class BOOST::vector < Float64>	Collection of forward DS imaginary parts
revDS_reals[ ][ ][ ]	Class BOOST::vector < Float64>	Collection of reverse DS real parts
revDS_imags[ ][ ][ ]	Class BOOST::vector < Float64>	Collection of reverse DS imaginary parts

2.1.2.8.2 ScienceDataProcessor Operations

Table 26 shows the science data processor operations.

Table 26. ScienceDataProcessor Operations

Name	Return	Parameters	
Cleans up of ScienceDataProcessor data for next dispatch.			
cleanup	primitive void	n/a	*n/a*
Populates high access frequency domain data.			
scalculateWavenumberBinSpacing	primitive	enum Band::Wavelength	*theBand*

Name	Return	Parameters	
	void		
Holds the intelligence to route any RDR packet to assistant class processing.			
processRawDataRecord	Class Spectra*	n/a	*n/a*
Reads external environment data from a file.			
processEnvironmentalData	primitive bool	typedef struct SYSTEMTIME&	*theSpectraFrameTime*
Builds polarization curve.			
buildPolarizationCurve	primitive void	Class Spectra*	*newSpectra*
		enum FieldOfRegard	*theScene*
		enum Band::Wavelength	*theband*
Creates and adds a new spectrnum to the spectra sliding window. Reference spectra are simply returned.			
AddSpectra	Class Spectra*	Class Spectra*	*newSpectra*
Builds each child table for every band and fov.			
refreshCorrectionMatrix	primitive int	primitive bool	*isLoading*
Multiplies child correction tables to calculate the final calibration matrix.			
calculateCalibrationMatrix	primitive void	n/a	*n/a*
cCombines requested corrections.			
combineRequestedCorrections	typedef unsigned int UInt32	typedef unsigned int UInt32	*requestedFOV*
Updates correction matrix parameters from configuration file.			
updateCorrectionMatrixParameters	primitive void	n/a	*n/a*
Calculates the hot radiance.			
calculateHotRadiance	Class Spectra*	enum SceneElement	*theFOV*
		enum Band::Wavelength	*theBand*
Calculates the cold radiance.			
calculateColdRadiance	Class Spectra*	enum SceneElement	*theFOV*
		enum Band::Wavelength	*theBand*
Sets the window size base on the spectral calibration method.			
setSpectralCalibrationMethod	primitive void	enum SpectralCalibrationMethod	*newMethod*
Removes a spectrum from the sliding window without adding a spectrum.			
flushWindow	Class Spectra*	enum SceneElement	*fov*
		enum Band::Wavelength	*band*
		enum FieldOfRegard	*scene*
		enum SweepDirection	*psd*
Processes Earth Scene spectra according to configuration parameters. Calibration, correction and validity are evaluated.			
processSpectra	Class Spectra*	Class Spectra*	*newSpectra*
Monitors the laser diode to determine if the CMO needs to be updated. This method also update the SDR Monitored Laser Wavelength and SDR Spectral Resampling Laser Wavelength.			
monitorLaserDiode	primitive bool	Class Spectra*	*theSpectra*
Determines if the laser wavelength needs to be updated and updates if needed.			
monitorLaserWavelength	primitive bool	n/a	*n/a*
Propagates temporal window size to assistant classe.			
setWindowSize	primitive	primitive int	*newSize*

Name	Return	Parameters	
	void		
Propagates temporal window size to assistant classes.			
declareWindowSize	primitive void	typedef unsigned int UInt32	*referenceSize*
		typedef unsigned int UInt32	*targetTempSize*
		typedef unsigned int UInt32	*calibrationSize*
Loads each correction matrix from a DMS.			
LoadCMOs	primitive bool	n/a	*n/a*
Calculates desired band center and configures spectra's desired band center.			
calculateDesiredBandCenter	primitive void	n/a	*n/a*
Applies the CMO to a spectrum.			
applyCMOCorrection	primitive void	Class Spectra*	*newSpectra*
		enum SceneElement	*theFov*
		enum Band::Wavelength	*theBand*
Saves serialized CMO. The values are from correction matrix class serialized.			
saveSerializedCorrectionMatrix	primitive void	n/a	*n/a*
Loads serialized CMO from DMS and populates correction matrix class.			
loadSerializedCorrectionMatrixFromDMS	primitive void	n/a	*n/a*
Computes field of views line-of-sight in spacecraft body frame.			
calculateGeometricCalibration	Class Spectra*	Class Spectra*	*newSpectra*
Refreshes a spectrum time from telemetry data.			
refreshTime	primitive void	Class CCSDSFormat*	*theTelemetryFormat*
Synchronizes all windows based on reference spectra sequence counts and updates sequences counts.			
syncWindows	primitive void	n/a	*n/a*
Calls the Spectra Manager's batchReferenceFCE which does fringe count detection and correction.			
launchBatchReferenceDetect	primitive void	Class SpectraManager::ReferenceSpectra	*spectraType*
		enum SweepDirection	*thePSD*
Interpolates a subset such that individual offset values are available for each 8 seconds scan period.			
buildBaffleTemperatureOffset	primitive void	Class BOOST::vector<Float32>&	*source*
		typedef int Int32	*orbitDuration*
Calculates the gain introduced by the FIR Filter.			
buildFirFilterGainTable	primitive void	enum Band::Wavelength	*requestedBand*
		typedef int Int32	*foldIndex*

2.1.2.9 Class SpectraManager

This class is responsible for each Spectra currently used in the moving window average. Table 27 shows the SpectraManager attributes and Table 28 shows the SpectraManager operations.

2.1.2.9.1 SpectraManager Attributes

Table 27. SpectraManager Attributes

Name	Type	Description
earthSceneTemp_Q[ ][ ][ ]	Class BOOST::vector<std::complex<F	*Collection of deep space spectra*

Name	Type	Description
	loat64> >	
hotColdDiff[ ][ ][ ]	Class BOOST::vector<std::complex<Float64> >	*Collection of differences between hot and col spectra*
windowSize	typedef int Int32	*Window size*
monitorWindowTime	primitive bool	*Monitor window time*
measuredSpectra[ ][ ][ ][ ]	typedef list<Spectra*> SpectraCollection	*Collection of measured spectra*
streamReference	typedef list<Spectra*> SpectraCollection	*Stream reference*
referenceSpectra[ ][ ][ ][ ]	Class Spectra*	*Collection of reference scenes*
windowSize	primitive int	*Depth of reference average*
SpectraCollection	typedef SpectraCollection	*Collection of measured scenes*
referenceSpectra[ ][ ][ ][ ]	Class Spectra*	*Collection of calculated average scenes*
hotReferenceFieldOfRegard	enum FieldOfRegard	*Hot reference field of regard*
coldReferenceFieldOfRegard	enum FieldOfRegard	*Cold reference field of regard*
refWindowSize	typedef int Int32	*Reference window size*
calWindowSize	typedef int Int32	*Calibration window size*
valWindowSize	typedef int Int32	*Valid window size*
maintainReferenceSceneVariance	primitive bool	*Reference scene variance*
correctionFilename[ ][ ]	typedef list<string> CMOCollection	*Collection matrix file names*
polarizationSpectra[ ][ ]	typedef SpectraCollection	*Collection of polarization spectra*
windowTimeSpan[ ][ ][ ][ ]	typedef struct SYSTEMTIME	*Collection of window time*
radiance[ ][ ][ ]	Class Spectra*	*Collection of radiance spectra*
rawSummationSpectra[ ][ ][ ][ ]	Class Spectra*	*Collection of raw spectra summation*
lastSpectra[ ][ ][ ][ ]	Class Spectra*	*Collection of last spectra*
currentCorrectionFilename[ ][ ]	primitive string	*Collection of current file name*
currentPolarizationSpectra[ ][ ]	Class Spectra*	*Collection of current polarization spectra*
geolocationSpectra[ ][ ]]	typedef list<Spectra*> SpectraCollection	*Collection of current geolocation spectra*
rawSummationSpectraNoCal[ ][ ][ ][ ]	Class Spectra*	*Collection of no calibrated spectra summation*
squareSummationSpectraNoCal[ ][ ][ ][ ]	Class Spectra*	*Collection of no calibrated spectra square summation*
trackLunarIntrusion	primitive bool	*Lunar intrusion flag*
maxLunarIntrusionRatio	typedef float Float32	*Maximum lunar intrusion ratio*
edrMinimumWavenumber[ ]	typedef double Float64	*EDR minimum wavenumber*
edrMaximumWavenumber[ ]	typedef double Float64	*EDR maximum wavenumber*
lunarIntrusionCount[ ][ ][ ]	typedef unsigned int UInt32	*Collection of lunar intrusion counts*
requiresPhaseSync[ ][ ][ ]	primitive bool	*Collection of spectra phase sync*
lastReferenceSpectra[ ][ ][ ][ ]	Class Spectra*	*Collection of last reference spectra*
holdingReferenceSpectra[ ][ ][ ][ ]	Class Spectra*	*Collection of holding reference spectra*
LinearityCorrectionParameter[ ]	typedef struct LinearityCorrectionParameters	*Collection of linearity correction parameters*
FIR_GAIN[ ]	Class Spectra*	*Collection of FIR gain*
performLinearityCorrectionControl[ ]	primitive bool	*Collection of information for performing linearity correction*
boxcarAverageWidth	typedef int Int32	*smoothing width for NEdN Estimate*

2.1.2.9.2 SpectraManager Operations

Table 28. SpectraManager Operations

Name	Return	Parameters	
Cleans up all obsolete spectra.			
Cleanup	primitive void	n/a	*n/a*

Name	Return	Parameters	
Initializes data for re-tasking.			
Init	primitive void	n/a	*n/a*
Adds a correction spectrum to the moving window for synchronized application to calibration.			
scheduleCorrection	primitive void	primitive string	*newCMOFileName*
		enum SceneElement	*theFov*
		enum Band::Wavelength	*theBand*
Adds the newSpectra to the measure spectra container.			
addSpectra	Class Spectra*	Class Spectra*	*newSpectra*
Adds hot and cold target curves to the radiance container for the current spectra.			
addTargetRadianceCurves	primitive void	Class Spectra*	*hotTargetCurve*
		Class Spectra*	*coldTargetCurve*
Synchronizes the windowed data in the event the data set does not begin on an 8 sec boundary.			
syncReferenceSpectra	primitive bool	Class Spectra&	*targetSpectra*
		primitive bool	*performFringeCountErrorHandling*
Reconciles/Synchronizes the reference spectra's phase to the target spectra's phase.			
reconcileReferenceSpectraPhase	primitive bool	Class Spectra&	*targetSpectra*
		typedef int Int32	*maxFCETries*
Returns requested measured spectra.			
getMeasuredSpectra	Class Spectra*	typedef int Int32	*temporalItem*
		enum SceneElement	*fov*
		enum Band::Wavelength	*band*
		enum FieldOfRegard	*scene*
		enum SweepDirection	*psd*
Returns requested correction spectra (mathematical correction matrix).			
getCorrectionFilename	primitive string	typedef int Int32	*temporalItem*
		enum SceneElement	*theFov*
		enum Band::Wavelength	*theBand*
Gets the valid count of spectra for a given fov, band, scene, and PSD			
getValidCount	typedef int Int32	enum SceneElement	*fov*
		enum Band::Wavelength	*band*
		enum FieldOfRegard	*scene*
		enum SweepDirection	*psd*
Sets the internal window size. If the content of the current window exceeds the new size, the oldest contents are discarded.			
setWindowSize	primitive void	typedef int Int32	*numberOfSpectra*
Forces a Spectra out of the window without adding one.			
purgeSpectra	Class Spectra*	enum SceneElement	*theFov*
		enum Band::Wavelength	*theBand*
		enum FieldOfRegard	*theFOR*
		enum SweepDirection	*thePSD*
Maintains Slope and Intercept values for rapid mathematical use.			
buildNoisePrediction	buildNoisePrediction	enum SceneElement	*theFov*
		enum Band::Wavelength	*theBand*
		enum SweepDirection	*thePSD*
Propagates temporal window size to assistant classes.			
declareWindowSize	typedef unsigned int UInt32	typedef unsigned int UInt32	*referenceSize*
		typedef unsigned int UInt32	*calibrationSiz*
Marks the calibration window invalid for current contents.			
resetCalibrationWindow	primitive void	enum Band::Wavelength	*band*
		typedef unsigned int UInt32	*newFoldIndex*
Returns the total count of all buffering windows requested.			
getBufferedSpectraCount	typedef unsigned int UInt32	primitive bool	*includeReferenceCount*
Adds a polarization correction spectra to the moving window for synchronized application.			

Name	Return	Parameters	
schedulePolarizationCorrection	primitive void	Class Spectra*	*newSpectra*
		enum FieldOfRegard	*patternScene*
		enum Band::Wavelength	*patternBand*
Returns requested polarization correction spectra.			
getPolarizationSpectra	Class Spectra*	typedef int Int32	*temporalItem*
		enum FieldOfRegard	*theScene*
		enum Band::Wavelength	*theBand*
Adds a geolocation spectrum to the moving window for synchronized application to calibration.			
scheduleGeolocation	primitive void	Class Spectra*	*newSpectra*
		enum FieldOfRegard	*theScene*
		enum SceneElement	*theFov*
Returns requested geolocation spectra.			
getGeolocationSpectra	Class Spectra*	typedef int Int32	*temporalItem*
		enum FieldOfRegard	*theScene*
		enum SceneElement	*theFov*
Calibrates the newMeasurement and performs a ratio threshold check against the corresponding window average.			
monitorLunarIntrusion	primitive bool	Class Spectra&	*newMeasurment*
Adjusts window phase.			
adjustWindowPhase	typedef int Int32	enum FieldOfRegard	*theFOR*
		enum SceneElement	*theFov*
		enum Band::Wavelength	*theBand*
		enum SweepDirection	*thePSD*
		Class BOOST::vector< std::complex<Float64> >&	*linearPhaseShift*
Removes a given spectra from the rawSummationSpectra and squareSummationSpectra.			
removeInvalidSpectra	primitive void	Class Spectra*	*oldSpectra*
		enum SpectralCalibrationMethod	*calType*
Starts a batch processing job on holdingReferenceSpectra. Detects FCE on each Ref Scene starting with LW1. If LW1 does not detect we move on to LW2, and so on and so forth. When a detection passes that fringe count is applied to each fov, band, and PDS of that FOR. The ref scene is added to the window.			
batchReferenceFCE	primitive void	enum ReferenceSpectra	*spectraType*
		enum SweepDirection	*psd*
Adds new reference spectra to holdingReferenceSpectra container.			
addHoldingReferenceSpectra	primitive void	Class Spectra*	*newSpectra*
ICT and DS Synchronization batch job that is done when the first ES pops out of the window.			
batchICTDSSynchronization	primitive bool	Class Spectra*	*oldSpectra*
		primitive int	*maxFCETries*
Starts a batch processing job of Earth Scene Spectra. The argument oldSpectra is the spectra that just popped out of a window for a certain FOR and PSD. Detection is started on LW1, and continues for each FOV and Band until a fringe count passes or all FOVs and Bands are exhausted.			
batchEarthSceneFCE	primitive bool	Class Spectra*	*oldSpectra*
Calculates the estimated detector voltage at the preamp.			
calculateVdc	primitive bool	Class Spectra*	*sceneSpectra*

2.1.2.10 Class Spectra

This class is the container for the processed video data. Table 29 shows the Spectra attributes and Table 30 shows the Spectra operations.

2.1.2.10.1 Spectra Attributes

Table 29. Spectra Attributes

Name	Type	Description
response	typedef double Float64	Response flag

realWavenumberBin	Class BOOST::vector<double>	Collection of real component of complex spectra
imaginaryWavenumberBin	Class BOOST::vector<double>	Collection of imaginary component of complex spectra
NEdN_Estimate	Class BOOST::vector<Float64>	Collection of NEdN estimation
firstWavenumber	typedef double Float64	Frequency of amplitude in bin zero
wavenumberBinSize	typedef double Float64	Frequency offset per bin (deltaSigma)
getVersionNumber	typedef unsigned short int	Version number
foldIndex	typedef unsigned int UInt32	Index
fceParameters[ ]	typedef struct FCEParam	Collection of FCE parameters
spectralBand	enum Band::Wavelength	Actual band reported in interferogram
fieldOfView	enum SceneElement	Actual field of view reported in interferogram
fieldOfRegard	enum FieldOfRegard	Actual field of regard reported in interferogram
sweepDirection	enum SweepDirection	Actual porch swing direction reported in interferogram
userLocks	typedef int Int32	User lock flag
scalingEnabled	primitive bool	Scaling enabled flag
Channel	typedef int Int32	Channel number
RDR_Status	typedef struct RawDataRecordStatusRegister	Collection of interferogram status information
saveVersion400	primitive void	Version number

2.1.2.10.2 Spectra Operations

Table 30. Spectra Operations

Name	Return	Parameters	
References to the root of the complex squares.			
getMagnitude	Class BOOST::vector<Float64 >&	n/a	*n/a*
Resizes the bin count.			
setSize	primitive void	typedef unsigned int UInt32	*newSize*
Clears internal data vectors.			
clear	primitive void	n/a	*n/a*
Adds complex squares.			
addSquare	primitive void	Class Spectra&	*source*
		primitive bool	*complex*
Removes complex squares.			
removeSquare	primitive void	Class Spectra&	*source*
		primitive bool	*complex*
Divides each element of the spectra by a specified number.			
operator/	primitive void	typedef double Float64	*factor*
Multiplies each element of the spectra by a specified number.			
operator*	primitive void	typedef double Float64	*factor*
Multiplies each element of the spectra bin by bin with the factor.			
operator*	primitive void	Class Spectra&	*factor*
Divide the spectra by a specified spectra.			
operator/	primitive void	Class Spectra&	*divisor*
Makes a copy of source spectra.			
operator=	Class Spectra&	Class Spectra&	*source*
Adds a spectrum to the current spectra.			
operator+	primitive void	Class Spectra&	*rightOperand*
Subtracts a spectrum from the current spectra.			

Name	Return	Parameters	
operator-	primitive void	Class Spectra&	*rightOperand*
Returns packed content of the class.			
save	primitive string	n/a	*n/a*
Clips and re-samples it's self to new resolution.			
resample	primitive bool	typedef double Float64	*startWavenumber*
		typedef double Float64	*stopWavenumber*
		typedef double Float64	*binSize*
Clips and re-samples it's self to new resolution.			
clipGuardBands	primitive bool	typedef double Float64	*startWavenumber*
		typedef double Float64	*stopWavenumber*
Returns packed content of the class.			
saveVersion400	primitive void	primitive string&	*binaryImage*
Generates NEdN estimations.			
generateUSN	primitive void	Class Spectra*	*rawSum*
		Class Spectra*	*squaredSum*
		typedef int Int32	*windowSize*
Smoothes the spectra.			
Smooth	primitive void	typedef int Int32	*windowSize*
Applies linearity error correction to the spectra.			
applyLinearityErrorCorrection	primitive void	n/a	*n/a*
Updates CrIS SDR geolocation.			
updateCrisSdrGeolocation	primitive void	typedef struct CrisSdrAlgDataType*	*algDataPtr*
		typedef int Int32	*scanIdx*
Updates CrIS SDR data.			
updateCrisSdrData	primitive void	typedef struct CrisSdrAlgDataType	**sdrPtr*
		typedef struct CrisSdrHdrDataType	**hdrPtr*
		typedef unsigned int UInt32	*scanIdx*
		std::auto_ptr<ScienceDataProcessor>&	*ScienceDataProcessorPtr*

2.1.2.11 Class TelemetryProcessor

This class provides the interface to retrieve science data from raw instrument telemetry. It encapsulates the extraction of housekeeping, diagnostic and normal mode interferograms. Table 31 shows the TelemetryProcessor attributes and Table 32 shows the TelemetryProcessor operations.

2.1.2.11.1 TelemetryProcessor Attributes

Table 31. TelemetryProcessor Attributes

Name	Type	Description
LRV	typedef TelemetryRecord	*Last received value*
apidInfoMap	typedef std::map<UInt16, ApidInfoStruct*> ApidInfoMap	*Collection of apid information*
apidCollection	typedef std::vector<UInt16> ApidCollection	*Collection of apid information*
ValueToChannelMap	typedef std::map<Int32, UInt32> ValueToChannelMap	*Collection of value to channel information*
channelToDetector[ ][ ]	typedef unsigned int UInt32	*Collection of channel to detector*
detectorToChannel[ ][ ]	typedef unsigned int UInt32	*Collection of detector to channel*
lastFrameTime	typedef struct SYSTEMTIME	*Last frame time*
lastFrameMiliTime	typedef unsigned int UInt32	*Last frame mili time*
lastFrameMicroTime	typedef unsigned short int UInt16	*Last frame micro time*

lastFrameDayTime	typedef unsigned short int UInt16	*Last frame day time*
telemetryLookup	Class map<string, Int32>	*Collection of telemetry lookup data*
fourMinPktUpdateIndex	typedef int Int32	*Four minutes package update index*
theCurrentVideoSource	enum BurstModes	*Collection of burst modes*
theVideoMode	enum BurstModes	*The video mode*
theEndOfEightSecEpoch	primitive bool	*The end of eight seconds epoch*
purgeCount	typedef int Int32	*The purge count*
theTelemetryVersion	enum TelemetryVersions	*Collection of telemetry versions*
fullVideoMode	primitive bool	*Full video mode flag*

2.1.2.11.2 TelemetryProcessor Operations

Table 32. TelemetryProcessor Operations

Name	Return	Parameters	
Returns index of first telemetry point with the specified APID.			
findRecord	typedef int Int32	typedef unsigned short int UInt16	*targetFrameType*
findRecord	typedef int Int32	primitive string	*targetName*
makePoint	primitive void	primitive string&	*pointName*
		primitive string&	*pointDescription*
		primitive string&	*subsystem*
		primitive string&	*frameGroup*
		typedef unsigned short int UInt16	*frameType*
		Class TelemetryPoint::TelemType	*telemType*
		typedef unsigned int UInt32	*bitLocation*
		typedef unsigned int UInt32	*pointSize*
		primitive string&	*pointUnits*
		typedef unsigned int UInt32	*telemetryPointID*
		Class TelemetryPoint::CoefContextCollection*	*coefficientContexts*
		Class TelemetryPoint::LimitContextCollection*	*limitContexts*
Returns a constant reference to the telemetry point at the given index.			
getRecord	Class TelemetryPoint&	typedef int Int32	*vectorIndex*
Clears and performs cleanup operations on the telemetry record.			
purgeTelemetryRecord	primitive void	n/a	*n/a*
Returns the name of the frame group to which the specified APID belongs.			
identifyFrameType	primitive string	typedef unsigned short int UInt16	*keyAPID*
Parses a single telemetry frame.			
parseDisplayData	primitive void	typedef unsigned short UInt16	*theAPID*
		typedef struct SYSTEMTIME	*&theTime*
		typedef unsigned int UInt32	*theMiliTime*
		typedef unsigned short UInt16	*theMicroTime*
		typedef unsigned short UInt16	*theDayTime*
		typedef unsigned char UInt8	**newData*
Extracts the video data from the supplied data.			
extractVideoData	primitive void	typedef unsigned int UInt32	*detector*
		typedef unsigned int UInt32	*wavelength*
		typedef unsigned char UInt8	**trimmedData*
Extracts the diagnostic data from the supplied data.			
extractDiagnosticData	primitive void	typedef unsigned short UInt16	*theAPID*
		typedef unsigned int UInt32	*detector*
		typedef unsigned int UInt32	*wavelength*
		typedef unsigned char UInt8	**detData*
		typedef int Int32	*dataOffset*
Matches the specified APID with a detector and a wavelength.			

Name	Return	Parameters	
rdrLookup	primitive void	typedef unsigned short UInt16	*theAPID*
		typedef unsigned int UInt32	**detector*
		typedef unsigned int UInt32	**wavelength*
		typedef unsigned int UInt32	**scene*
Resets LastBurstApidDetection by forgetting that any of the APIDs were in the last burst.			
resetLastBurstApidDetection	primitive void	n/a	*n/a*
Updates TelemetryPoint stale value.			
updateStaleValues	primitive void	n/a	*n/a*
Checks whether the specified APID is recent.			
isApidRecent	primitive bool	typedef unsigned short UInt16	*theApid*
Finds and returns the info associated with the specified apid.			
findApidInf	typedef struct ApidInfoStruct*	typedef unsigned short UInt16	*apid*
Reads telemetry points data.			
readTelemetryPoints	primitive void	n/a	*n/a*
Checks if telemetry record defined.			
enableLimitChecking	primitive void	primitive bool	*limitCheck*

2.1.2.12 Class VideoData

This class is responsible for internal bit-trim reversal behavior of the TelemetryProcessor class. Table 33 shows the VideoData attributes and Table 34 shows the VideoData operations.

2.1.2.12.1 VideoData Attributes

Table 33. VideoData Attributes

Name	Type	Description
theExtractionRecord[ ]	struct EXTRACTION_RECORD	Collection of band specific bit trim values
theTlmLookupTable[ ]	struct TELEMETRY_LOOKUP	Collection of complex samples

2.1.2.12.2 VideoData Operations

Table 34. VideoData Operations

Name	Return	Parameters	
Populates the internal symbol table used to unpack and extract the bit-trimmed data set.			
initSymbolTable	primitive void	primitive bool	*useDefaultTable*
Coordinates the real and imaginary extraction of the complex samples into the real and imaginary portions.			
Extract	primitive void	typedef unsigned int UInt32	*detector*
		typedef unsigned int UInt32	*wavelength*
		typedef unsigned char UInt8	**trimmedData*
Retrieves the correct trim value from the symbol table for a given wavelength sample.			
getTrimSize	typedef unsigned short UInt16	typedef unsigned int UInt32	*wavelength*
		typedef int Int32	*sample_index*
Established the 4 minute bit trim tables for all three bands.			
buildExtractionTables	primitive void	primitive bool	*useDefaultTable*
Extracts the original sample from the pack data stream by using the bit-trimming information configured in the symbol table.			
restoreSamples	primitive void	typedef unsigned int	*detector*

Name	Return	Parameters	
		UInt32	
		typedef unsigned int UInt32	*wavelength*
		typedef unsigned short int UInt16	*dataElement*
		primitive bool	*realComponent*

2.1.2.13 Class ScienceCalibrationRecord

This class is responsible for processing the eight second science calibration packet (Refer to the Equation (79e) in Section 5.5 Temperature Computation, BOM-CrIS-0067 Revision E. The constants 1, 1, 2 and 5 are the coefficients of the Taylor series expansion (Note: ITT modifies these coefficients in V2.14.0)). It maintains the sliding window average ICT temperature. Table 35 shows the ScienceCalibrationRecord attributes and Table 36 shows the ScienceCalibration operations.

2.1.2.13.1 ScienceCalibrationRecord Attributes

Table 35. ScienceCalibrationRecord Attributes

Name	Type	Description
CalibrationParameter	typedef struct CalibrationParam	Collection of calibration configuration values
TestEnvironmentParameter	typedef struct CrIS_TestEnvironmentParameters	Collection of instrument configuration values
GeneralParameter	typedef struc SDR_GeneralParameters	Collection of processing configuration values
AlgorithmParameter	typedef struct SDR_AlgorithmnParameter	Collection of algorithm configuration values
processingIctPeriod	Class TemperatureHistory	Collection of temperature measurements
processingIctSensor1	Class TemperatureHistory	Collection of temperature measurements
processingIctSensor2;	Class TemperatureHistory	Collection of temperature measurements
processingBeamsplitterPeriod	Class TemperatureHistory	Collection of temperature measurements
processingScanMirrorPeriod	Class TemperatureHistory	Collection of temperature measurements
processingOmaPeriod	Class TemperatureHistory	Collection of temperature measurements
processingScanBafflePeriod	Class TemperatureHistory	Collection of temperature measurements
processingTelescopePeriod	Class TemperatureHistory	Collection of temperature measurements
processingCoolerStage1Period	Class TemperatureHistory	Collection of temperature measurements
processingCoolerStage2Period	Class TemperatureHistory	Collection of temperature measurements
processingCoolerStage3Period	Class TemperatureHistory	Collection of temperature measurements
processingCoolerStage4Period	Class TemperatureHistory	Collection of temperature measurements
processingLaserDiodeCurrentPeriod	Class TemperatureHistory	Collection of temperature measurements
processingLaserDiodeTempPeriod	Class TemperatureHistory	Collection of temperature measurements
processingPeriod	Class TemperatureHistory	Collection of temperature measurements
ieCCACalibResistorTemp[ ]	typedef int Int32	Collection of resistor temperature measurements
lowRangeCalibResistor[ ]	typedef int Int32	Collection of low range resistor measurements
highRangeCalibResistor[ ]	typedef int Int32	Collection of high range resistor measurements
ictTemp[ ][ ]	typedef int Int32	Collection of ICT temperature measurements
crossTrackServoErr[ ]	typedef int Int32	Collection of cross track servo error measurements

inTrackServoErr[ ]	typedef int Int32	Collection of in track servo error measurements
laserDiodeCurrent	typedef int Int32	Laser diode current
laserDiodeTemp	typedef int Int32	Laser diode temperature
beamsplitterTemp	typedef int Int32	Beam splitter temperature
omaStructureTemp1	typedef int Int32	OMA structure temperature
omaStructureTemp2;	typedef int Int32	OMA structure temperature
ssmScanMirrorTemp;	typedef int Int32	Scan mirror temperature
ssmScanMirrorBaffleTemp	typedef int Int32	Scan mirror baffle temperature
coolerStage1Temp	typedef int Int32	Cooler stage1 temperature
coolerStage2Temp	typedef int Int32	Cooler stage2 temperature
coolerStage3Temp	typedef int Int32	Cooler stage3 temperature
coolerStage4Temp	typedef int Int32	Cooler stage4 temperature
telescopeTemp;	typedef int Int32	Telescope temperature
captureNextLaserInfo	primitive bool	Next laser information
capturedLaserDiodeCurrent	typedef double Float64	Laser diode current
capturedLaserDiodeTemperature	typedef double Float64	Laser diode temperature
previousAveragedLaserDiodeTemp	typedef double Float64	Previous averaged diode temperature
previousAveragedLaserDiodeCurrent	typedef double Float64	Previous averaged diode current
servoValuesChanged	primitive bool	Servo value change flag
convertedCrossTrackServoErr[ ]	typedef double Float64	Converted cross track servo error
convertedInTrackServoErr[ ]	typedef double Float64	Converted in track servo error

2.1.2.13.2 ScienceCalibrationRecord Operations

Table 36. ScienceCalibration Operations

Name	Return	Parameters	
Calls clearHistory to free memory.			
clearHistory	primitive void	n/a	*n/a*
Sets window size.			
setWindowSize	primitive void	typedef int Int32	*newSize*
Reads Values from Science Telemetry Packet into class variables.			
refreshData	primitive bool	typedef unsigned short UInt16	*apid*
		typedef struct CrisSdrAlgDataType*	*algDataPtr*
		typedef struct IngMsdCoefficients_CrisSdrStruct*	*cfgParmsPtr*
		typedef unsigned int UInt32	*scanIdx*
Calculates ICT temperature and adds to sliding window average.			
calculateIctTemperature	primitive void	Class TemperatureHistory::AvgPRTTemp	**avgIctTempPtr*
		typedef struct IngMsdCoefficients_CrisSdrStruct*	*cfgParmsPtr*
Calculates Beamsplitter temperature and adds to sliding window average.			
calculateBeamsplitterTemperature	typedef double Float64	n/a	n/a
Calculates Scan Mirror temperature and adds to sliding window average.			
calculateScanBaffleTemperature	typedef double Float64	n/a	n/a
Calculates OMA temperature and adds to sliding window average.			
calculateOmaTemperature	typedef double Float64	n/a	n/a
Adds a missing temperature record to the temperature history window.			
markMissingCal	primitive	n/a	n/a

Name	Return	Parameters	
	void		
Removes last temperature record from the temperature history window.			
removeLastCal	primitive void	n/a	n/a
Finds the max and min temperatures in the temperature history window and determines if the difference exceeds the Science Telemetry Drift Limits.			
hasExcessThermalDrift	primitive bool	n/a	n/a

2.1.2.14 SDR Generator Application COTS Components

2.1.2.14.1 uBLAS (BOOST)

BOOST version 1.32.0, uBLAS provides templated C++ classes for elementary linear algebra, and access into vectors and matrices by way of matrix and vector adapters.

2.1.2.14.2 Serialization (BOOST)

CrIS SDR uses the BOOST version 1.32.0 Serialization library to read and write objects to and from memory allocated by DMS.

2.1.2.14.3 LAPACK

CrIS SDR uses the LAPACK version 3.0 dgetrf and dgetri functions when inverting matrices.

2.1.2.14.4 FFTW

CrIS SDR uses three one-dimensional discrete Fourier transform routines from FFTW version 3.0.1; one which performs DFT on real data returning only the real part of the result, another which performs DFT on real data and returns a complex result, and finally, a function that performs DFT on complex data and returns a complex result.

2.1.2.15 The Sliding Window

The C++, CrIS SDR code, converted from MATLAB by ITT, was designed to work with a continuous stream of data. The processing model used by NPOESS sends data to the CrIS SDR processes in discrete (not necessarily contiguous) chunks called granules. These granules are stored in data structures called RDRs.

CrIS granules contain one or more scans. As shown in Figure 3, each scan consists of four reference FORs and 30 earth view FORs. Each FOR is composed of nine FOVs which, in turn, are comprised of three wavelength bands (LW, MW, and SW). Each FOR/FOV/wavelength combination is processed independently and stored in separate sliding windows.

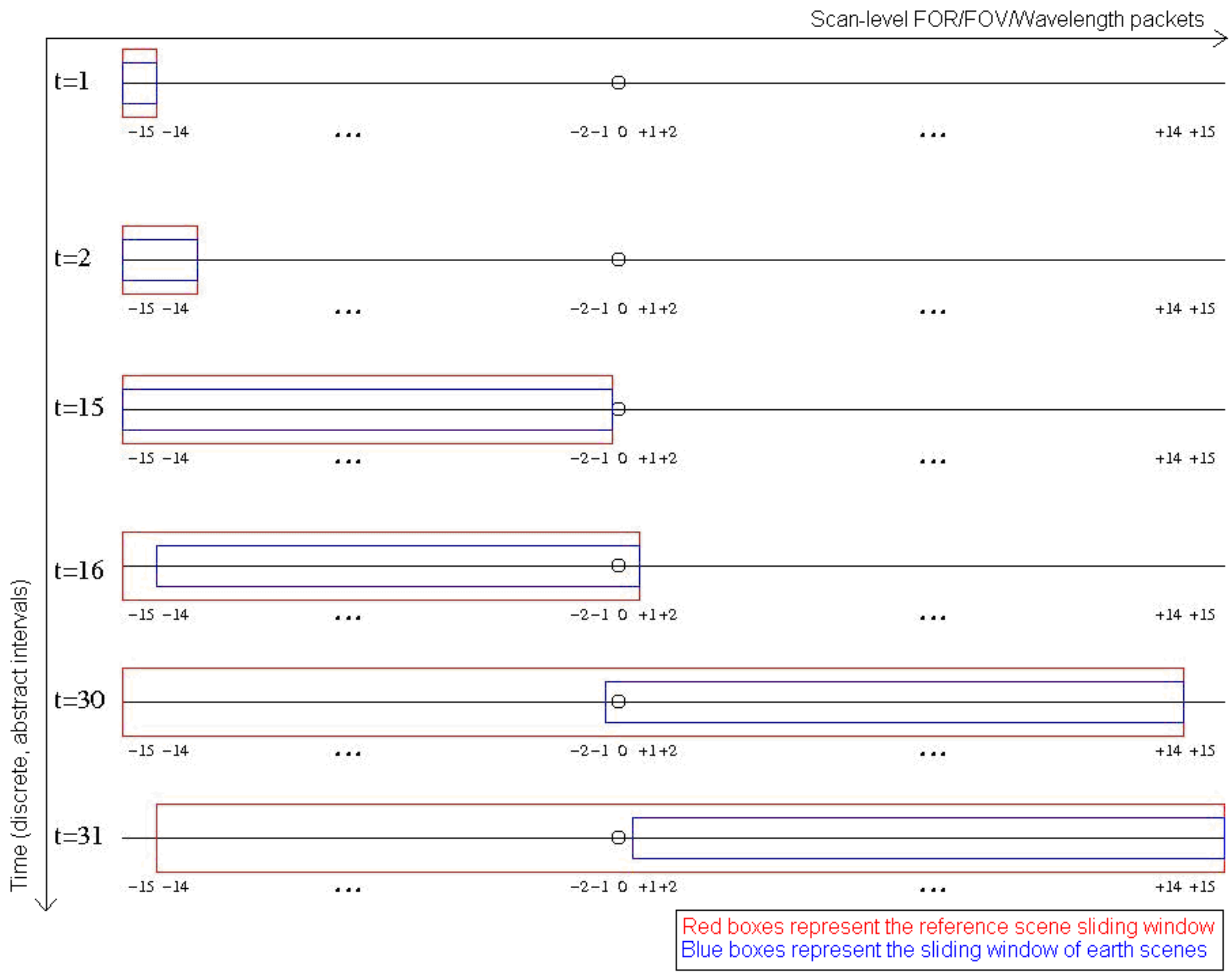

Figure 4. Sliding Window Concept

Here, the red box is the reference scene sliding window and the blue box is the sliding window of earth scenes. The figure shows the process of building the sliding windows from nothing and then processing one scan.

Because only the earth-view FORs are calibrated, there are 30x9x3 separate sliding windows. The code also differentiates between forward and reverse scans (hence the need for both forward and reverse DS and ICT reference FORs); however, each Earth view FOR is always associated with either forward or backward scan so this does not double the number of sliding windows in the program.

Earth view spectra are also stored in a smaller sliding window. However, this sliding window has no scientific value and is used as a matter of convenience to assist tracking when an Earth scene spectrum should be processed. Because these spectra are not stored for processing other Earth view scans, only future Earth views need to be saved in this smaller window for future processing. Earth view spectra are processed upon being ejected from their sliding window. In the following sections, sliding window refers to the window of reference scenes.

The CrIS SDR implementation has hard-coded the sliding window size at 30, although this parameter is configurable via a global header file. (Consequently, however, this does imply that reconfiguration of the parameter requires a recompilation of the CrIS SDR library.) The earth view sliding window is one-half that size (current scan plus 14 future scans). The process of adding a single scan to a sliding window is shown in Figure 4.

2.1.2.15.1 One Additional Spectra is Needed to Move the Window

All Earth view spectra that are calibrated using a partial sliding window are marked as being of degraded quality. Earth view spectra are processed when ejected from the sliding window. Therefore, not only must enough scenes be retrieved to keep the sliding window full during processing, enough scans must be retrieved to eject the tasked spectra from the window. For example, when processing with granules consisting of a single scan, 31 scans must be retrieved (15 prior, plus 14 after, plus the scan contained within the granule that has been tasked to be processed to fill the window, plus one additional scan to eject the desired processed Earth views). If granules containing four scans were being processing, then 34 scans would be required.

2.1.2.15.2 Unaligned Scans Require Retrieval of Yet Another Scan

CrIS SDR scans are not guaranteed to align with granules boundaries. In other words, the time associated with the beginning of a scan is not equal to the time associated with the beginning of a granule. This implies that the RDR data structure need not begin with the first spectra of the first scan of the granule that has been tasked to process.

This CrIS SDR implementation ensures that the last scan is retrieved in its entirety by requesting enough granules to obtain yet another scan. This ensures success if at least a part of an extra scan has been retrieved; there must be complete scans for all scans before the extra scan. CrIS SDR uses the ceilings of the following formulae to liberally ensure that the entirety of each scan retrieved associated with the granule that has been tasked to process, where W represents the sliding window size and S represents the number of scans per granule:

Number of past granules to retrieve: $\left\lceil \frac{\frac{W}{2}}{S} \right\rceil$

Number of future granules to retrieve: $\left\lceil \frac{\frac{W}{2}+S}{S} \right\rceil$.

2.1.2.15.3 Granule Version Ids

The granules containing science data have a version id as a part of their meta-data. Occasionally, an RDR needs to be updated; and when this occurs, its granule version ID is modified as well.

Although not all scans are processed, all granules needed for processing are retrieved. Upon receipt of the granules needed for the latest tasking, CrIS SDR checks the version ID associated with the scans already in the sliding window from the last tasking. If one of those version ids has changed, then the entire sliding window is rebuilt from scratch.

2.1.2.16 Sliding Window Optimization

This implementation has ameliorated the latency introduced when switching from a stream-based input to a granule-based input paradigm by preserving as much of the sliding window as possible between taskings.

This is done by using granule IDs to determine how many granules have passed since the last tasking. CrIS SDR uses this value to determine how many future scans are added to the sliding window. If this difference implies that the number of passed scans is greater than the number of future scans already stored in the sliding window, the sliding window is cleared of all scans and refilled.

Figure 5 illustrates this concept in its idealized form. Four CrIS SDR processes are run concurrently, each of which processes a 4-scan granule. State A represents the current 4-process tasking, while State B represents the next 4-process tasking. There are 5 sequential tasks across the 2 different states, so 4 are processed in State A, while the last is processed in State B. Assume that Process 1 in State A is the first granule tasked to the CrIS SDR algorithm. This process fills the sliding window with 30 scans worth of data (15 past cross-granules scans, 4 scans from the tasked granule, and 11 future cross-granule scans) before processing the 4 tasked scans in the middle of the window. Each time a scan is processed, one scan is purged from the beginning of the window and the next scan from the future cross-granules is added to the end of the window. Therefore, 16 scans are purged while 16 are added from State-to-State during ideal processing.

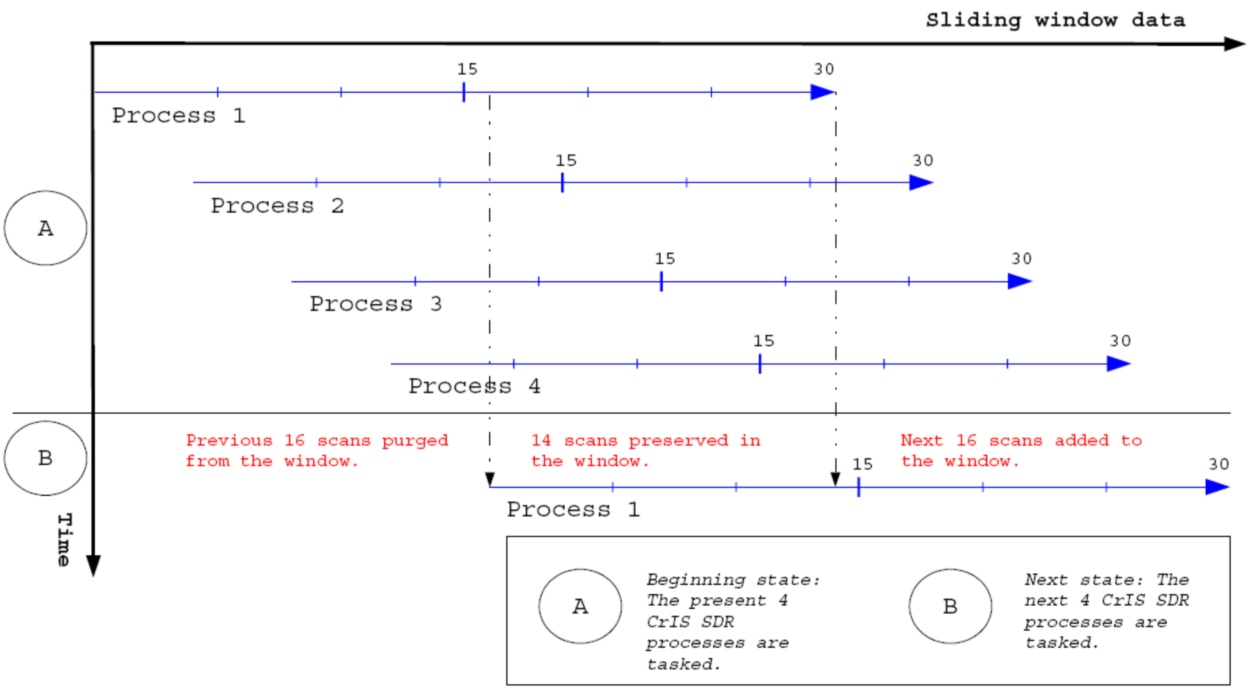

Figure 5. Sliding Window Optimization Concept

2.1.2.17 Creation and Storage of Correction Matrices

Parameters provided by the four-minute engineering packets can prompt CrIS SDR to rebuild its correction matrix. When a correction matrix rebuild is performed, both the new matrix and its corresponding four-minute packet are saved to DMS.

Adding only the scans needed to refill the sliding window limits the scans actually examined during processing. When tasked sequentially, or nearly sequentially, the four-minute engineering and eight-second packets enclosed within which the granule CrIS SDR is tasked to process, are not examined. This requires CrIS SDR to perform some correction matrix updates before they are to be used for processing.

Figure 6 shows an example of how a correction matrix build is handled in the case of sequential tasking. This figure shows us that the four-minute packet is added to the *end* of the sliding window *only once*. Processing any other granule results in that scan being placed at the +14th position. Thus CrIS SDR, when tasked sequentially, builds a correction matrix when a rebuild is detected for the +14th scan.

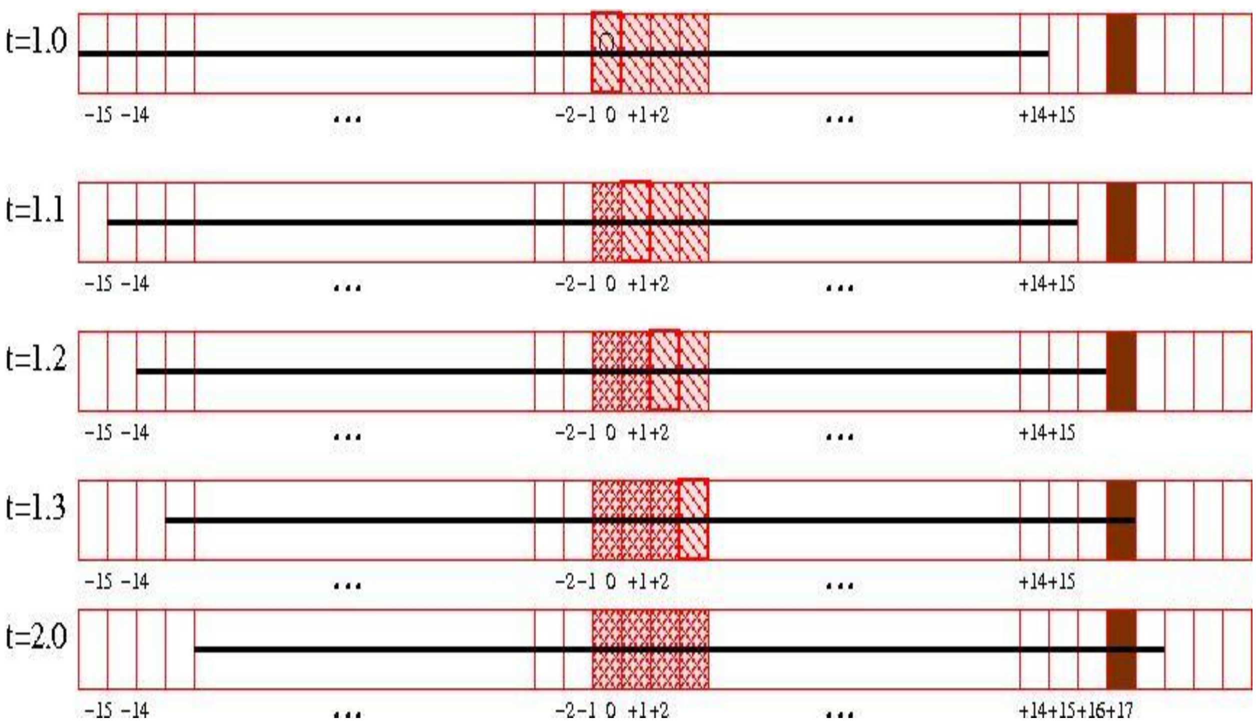

Figure 6. Sequential Tasking of a Correction Matrix Build

The boxes with single hashes all belong to the same, four-scan granule. The boxes with cross hashes are those scans after being processed. The scan currently being processed is surrounded by a bold border. The shaded box represents a scan which causes the correction matrix to be re-built. In this scenario, the correction matrix would be rebuilt at time 1.3. This scan is never again at the +14 location unless this granule is processed again.

The only other scenario where a correction matrix is rebuilt is when a rebuild is detected for the 0th (currently being processed) scan. This can only occur when the sliding window is being completely rebuilt. There is no concern about inadvertently building an already built matrix because the start time of the granule that caused the matrix rebuild is stored with the matrix as metadata. When tasked, this metadata is used to determine which correction matrix to load from DMS. When a scan that caused a rebuild during an earlier tasking (or even by another CrIS SDR process) is later than a part of the granule CrIS SDR is tasked to process, the previously built correction matrix and its accompanying four-minute engineering packet associated with this scan is the one that's returned by DMS. Therefore, parsing the four-minute engineering packet shows no change when compared to the engineering packet retrieved from DMS.

2.1.2.18 Performance

The processing of data is organized around an eight-second scan period, where the hot reference scene immediately precedes the eight-second science telemetry packet. The values derived from these packets remain constant for all measured earth scenes for the next eight-second period. The following items have been calculated at the beginning of this period.

- Hot and Cold variances for the moving window averages
- NEdN slope and intercept built from variance (optionally)
- Radiance for the current laser frequency

The sliding window average is maintained in a doubly-linked list where a sum and sum-of-squares is adjusted for each valid spectra entering or exiting the calibration period.

In the event of earth scene fringe count detection, each of the windowed Cold reference spectra are phase adjusted and the state of the average restored.

The 128 point Legendre polynomials used in the calculation of the ILS Self Apodization Matrix have been “hard-coded”.

When appropriate, the Correction matrix is loaded from DMS to address expensive recalculations that would occur at the receipt of the first four minute engineering packet.

2.1.2.19 ATBD - Operational Adaptation, Deviation or Limitations

The theoretical basis for the CrIS SDR Algorithm is documented in the following document:

D43773 Cross Track Infrared Sounder (CrIS) Sensor Data Records (SDR) Algorithm Theoretical Basis Document ATBD (ref BOM-CrIS-0067/ITT8180004).

The following deviations or adaptations to the referenced ATBD have been made in the C++ prototype code and are discussed in detail in this section.

1. Lunar Intrusion

Lunar Intrusion

Operationally the CrIS SDR Algorithm is required to support the detection and invalidation of space reference spectra that have experienced a lunar intrusion. The algorithm as described in the ATBD does not address this requirement, however, the final iteration of the algorithm’s prototype code described in later sections of this document does include functionality to detect and handle lunar intrusions.

The following steps are taken with a copy of each new Deep Space spectrum to detect a lunar intrusion:

1. Subtract the average DS spectrum from the new spectrum.
   $$\widetilde{S} = \widetilde{S}^{DS} - \left\langle \widetilde{S}^{DS} \right\rangle$$
2. Perform FCE handling on the new DS spectrum
3. Calculate the ratio of the new DS spectrum to the difference between the radiance of the ICT and Deep Space; DS radiance is non-zero for system level testing.

$$\tilde{L} = \frac{\tilde{\tilde{S}}}{B^{ICT} - B^{DS}}$$

4. Compute the average radiance per bin of the real component.

$$\frac{\int_{\sigma_{\min}}^{\sigma_{\max}} L}{\frac{(\sigma_{\max} - \sigma_{\min})}{\Delta\sigma} + 1}$$

σ_{min}, σ_{max} and $\Delta\sigma$ define the specified wavenumber bin scale for each band as expected by the EDR algorithm.

5. This value is roughly a ratio of the spectral radiance of the new scene as compared to the current Deep Space average. If the ratio is greater than 3%, Lunar Intrusion is assumed.

If Lunar Intrusion is detected, the new DS spectrum is marked as invalid and excluded from the Moving Window average. This action results in all Earth Scene spectra calibrated, while the invalid DS spectrum is in the Moving Window, to be marked as invalid with a flag identifying Lunar Intrusion. After the entire window is refreshed, i.e. the intruded spectrum falls out of the window after four minutes, Earth Scene spectra is considered valid.

2.1.3 Graceful Degradation

All CrIS SDR inputs are required. Therefore, there is no graceful degradation.

2.1.4 Exception Handling

Generally, exceptions are introduced by incompatible configuration setup for the data being processed. Handling of such events usually results in the offending spectra to be invalidated and processing continued according to the rules defined for missing data. The resultant log file contains details describing individual exceptions.

2.1.5 Data Quality Monitoring

The CrIS SDR algorithm provides for several assessments of the quality of given interferograms/spectra. There are several data quality flags that are set by the instrument at the interferogram data. Tables 37 through 41 describe the flags. For more information on quality flags, see the CDFCB-X.

Table 37. Band Level Quality Flags (Band 1)

Band Level Quality Flag B1	Bit	Value	Description and Handling
CrIS SDR Quality	0-1	0-2	This is a summary flag intended to indicate the SDR's radiometric quality. It is a pixel level 2-bit binary flag and is set for each band/FOV/FOR for a total of 1620 bits per scan.
Invalid Geolocation	2	0-1	If the flag was set, then the geolocation information included in the SDR is invalid. It is set for each band/FOV/FOR with a total of 810 bits per scan.
Invalid Radiometric Calibration++	3-4	0-2	This is a summary flag intended to indicate the SDR's radiometric quality. It is a pixel level 2-bit binary flag and is set for each band/FOV/FOR for a total of 1620 bits per scan.
Invalid Spectral Calibration++	5-6	0-2	This flag is intended to summarize the overall spectral calibration quality. This is a pixel level 2-bit binary flag and is set for each band/FOV/FOR for a total of 1620 bits per scan.

Band Level Quality Flag B1	Bit	Value	Description and Handling
Fringe Count Error Corrected	7	0-1	It is a pixel level binary flag (0 – successful FCE correction; 1 –failed FCE correction) and shall be set for each band/FOV/FOR with a total of 810 bits per scan.

Table 38. Band Level Quality Flags (Band 2)

Band Level Quality Flag B2	Bit	Value	Description and Handling
Day/Night Indicator	0	0-1	This flag is intended to identify day and night observations. This flag shall be set to 0 (day) if the local zenith angle is less than 90 degree, and to 1 (night) otherwise. It is a pixel level binary flag and is set for each band/FOV/FOR for a total of 810 bits per scan.
Invalid RDR Data	1	0-1	If this flag was set, then the instrument exhibited operational errors and the associated interferogram(s) is excluded from SDR processing. It is a pixel level binary flag and is set for each band/FOV/FOR with a total of 810 bits per scan.
Fringe Count Error Detect	2	0-1	If this flag was set, then a significant number of fringes have been missed, shifting the interferogram ZPD outside of a window monitored by the instrument, and the interferogram is excluded from SDR processing. The flag was set by the CrIS instrument and contained in the CrIS RDR data packets. It shall be a pixel level binary flag (0 – No Significant FCE Detected; 1 – Significant FCE Detected) and shall be set for each band/FOV/FOR with a total of 810 bits per scan.
Bit Trim Failed	3	0-1	This flag is intended to identify CrIS RDR interferograms that were clipped during the bit trimming process and are excluded from SDR processing. It is a pixel level binary flag and is set for each band/FOV/FOR with at total of 810 bits per scan.
Spare	4-7		Spare bits

Table 39. FOV Level Quality Flags

FOV Level Quality Flag	Bit	Value	Description and Handling
Lunar Intrusion Forward	0	0-1	This flag is intended to indicate that radiometric calibration could be of degraded quality due to moon in Deep Space view. This flag shall be set if DS counts are believed to be contaminated by moon disc radiance in any of the nine CrIS FOVs. It is a scan level binary flag and is set for each band/FOV/direction for a total of 27 bits per scan.
Lunar Intrusion Reverse	1	0-1	This flag is intended to indicate that radiometric calibration could be of degraded quality due to moon in Deep Space view. This flag shall be set if DS counts are believed to be contaminated by moon disc radiance in any of the nine CrIS FOVs. It is a scan level binary flag and is set for each band/FOV/direction for a total of 27 bits per scan.
Spare	2-7		Spare bits

Table 40. Scan Level Quality Flags

Scan Level Quality Flag	Bit	Value	Description and Handling
Data Gap	0	0-1	This quality flag checks RDR data for data gaps. This flag is set if there is a data gap, i.e. missing scan(s), preceding the current scan. It is a scan level binary flag and is set 1 bit per scan.
Timing Sequence Error	1	0-1	This flag is set if the recorded time is not in sequence. This is a granule level binary flag and is set 1 bit per scan.
Lambda Monitored Quality	2	0-1	This flag is intended to identify an invalid laser wavelength

Scan Level Quality Flag	Bit	Value	Description and Handling
			calculation due to invalid diode current and/or temperature measurements. This is a scan level binary flag and is set 1 bit per scan.
Invalid Instrument Temperatures	3	0-1	This quality flag is intended to identify the situation when the measured temperature of any instrument components (e.g., beam-splitter, scan mirror, scan baffle, etc.) are out of allowable ranges. This flag is a scan level 1-bit binary flag and is set for each scan for a total of 1 bit per scan.
Excess Thermal Drift	4	0-1	This flag is set to indicate at least one of the monitored instrument temperatures has drifted more than a specified tolerance value. This flag is a scan level 1-bit binary flag and is set for each scan for a total of 1 bit per scan.
SuspectNeonCalibration	5	0-1	This flag shall be set if 25% (tunable parameter) or more of the neon calibration dataset is rejected. It is a granule level binary flag (0 – accepted neon calibration; 1- rejected neon calibration) and has 1 bit per granule.
Spare	6-7		Spare bits

Table 41. Additional Quality Fields

Additional Quality Fields	Description and Handling
Impulse Noise Count	This flag represents the number of samples in an interferogram that exceeded the impulse noise mask and were set to zero; if > 1 the resultant spectrum is flagged as having excess noise. The flag is a pixel level 8-bits number flag and is set for each band/FOV/FOR with a total of 6480 bits per scan.
Deep Space Symmetry	This flag is intended to identify the asymmetry in the measured DS IGMs. This flag is a scan level 64-bit number flag and is set for each band/FOV/direction for a total of 1728 bits per scan. The value of this quality flag is computed using the following equation: $$DSS = \frac{1}{N}\sum_{i=1}^{N}\left\|S_{FWD}(i) - S_{REV}(i)\right\|$$ Where S_{FWD} and S_{REV} are the forward and reverse direction complex deep space view IGMs, respectively, and N is the length of the IGMs.
SDR FCE Count	This flag represents the FCE counts detected in the SDR processing.
Deep Space Window Size	This flag represents the number of DS spectra used to calibrate the earth scene. It is a scan-level, 16-bit number flag and is set for each band/FOV/direction for a total of 864 bits per scan.
ICT Window Size	This flag represents the number of ICT spectra used to calibrate the earth scene. It is a scan-level, 16-bit number flag and is set for each band/FOV/direction for a total of 864 bits per scan.
Deep Space Spectral Stability	This flag monitors the spectral variability of the DS views within the moving window. It is a scan level 64-bits number flag and is set for each band/FOV/direction for a total of 3456 bits per scan. It is calculated using the following equation: $$DSSS = \frac{1}{N}\sum_{i=1}^{N}\sigma(i)$$ Where N is the number of spectral bins, $\sigma(i)$ is the standard deviation of the DS spectra in the *i*th bin.
ICT Spectral Stability	This flag monitors the spectral variability of the ICT views within the moving window. It is a scan level 64-bits number flag and is set for each band/FOV/direction with a total of 3456 bits per scan. It is calculated using the following equation: $$ICTSS = \frac{1}{N}\sum_{i=1}^{N}\sigma(i)$$ Where N is the number of spectral bins, $\sigma(i)$ is the standard deviation of the ICT spectra in the *i*th bin.
ICT Temperature Stability	This flag measures the stability of the two Platinum Resistance Temperature measurements of the Internal Calibration Target. It is a scan level 32-bit number flag and is set for each of the two PRTs with a total of 64 bits per scan. This flag is calculated using the following equation: $$ICTTS = \sqrt{\sum_{i=1}^{ICTWS}[\overline{T}(i) - \frac{1}{ICTWS}\sum_{j=1}^{ICTWS}\overline{T}(j)]^2 /(ICTWS-1)}$$ Where ICTWS is the ICT Window Size, $\overline{T}(i) = \frac{1}{N}\sum_{k=1}^{N}T(i,k)$, where T(i,k) is the converted PRT temperature in the validated kth epoch, represents the average ICT temperature measured from a

Additional Quality Fields	Description and Handling
	PRT for the ith scan. This flag is set separately for the two PRTs.
ICT Temperature Consistency	This flag measures the consistency between the two Platinum Resistance Temperature measurements of the Internal Calibration Target. It is a scan level 32-bit number flag and has 32 bits per scan. This flag is calculated using the following equation: $$ICTTC = \frac{1}{ICTWS}\sum_{i=1}^{ICTWS}\overline{T}_1(i) - \frac{1}{ICTWS}\sum_{i=1}^{ICTWS}\overline{T}_2(i)$$ The nomenclature is similar to the ICT Temperature Stability flag except that the subscripts 1 and 2 denote the two PRTs, respectively.
Number of Valid PRT Temperatures	This flag represents the number of PRT temperatures that have been validated using the procedure described below. The mean and standard deviation of the PRT temperatures is calculated to identify any outliers, which are defined as being different from the mean by an amount of more than 3-sigma (standard deviation). The outliers do not enter subsequent processing. This is a scan level 8-bit number flag and is set for each of the two primary PRTs for a total of16 bits per scan.
CrIS RDR Yield	This flag represents the percentage of good quality RDRs in a granule. This shall be a granule level flag and has eight bits per granule for a total of 810 records per scan.
Measured Laser Wavelength	This quality flag represents the measured metrology laser wavelength with neon lamp calibration. It shall be a scan level 64-bit number flag and 64 bits per scan.
Lambda Monitored	This flag represents the monitored laser metrology wavelength, calculated using data from the 4-min engineering packets and Neon calibrated laser metrology wavelength. It is a scan level 64-bit number flag and has 64 bits per scan.
CrIS SDR Yield	This flag represents the percentage of good quality SDRs in a granule. It is a granule level flag and has eight bits per granule for a total of 810 records per scan.

2.1.6 Computational Precision Requirements

In general, all computations should be performed using double precision floating point arithmetic to maintain less than the required 1 ppm error contribution due to computation precision. It should be noted that the final output spectrum is represented with single precision floating point values.

2.1.7 Algorithm Support Considerations

Reference the list of libraries under Section 2.1.2.14.

2.1.8 Assumptions and Limitations

2.1.8.1 Assumptions

Currently this algorithm has only been tested with one scan granule.

2.1.8.2 Limitations

When the option to reassign the Field of Regard for the DS and/or ICT scenes to obtain calibrated output at one of these scenes, only one sweep direction of the two samples is calibrated based on which Earth Scene is selected as the hot reference.

3.0 GLOSSARY/ACRONYM LIST

3.1 Glossary

The current glossary for the NPOESS program, D35836_G_NPOESS_Glossary, can be found on eRooms. Table 42 contains those terms most applicable for this OAD.

Table 42. Glossary

Term	Description
Algorithm	A formula or set of steps for solving a particular problem. Algorithms can be expressed in any language, from natural languages like English to mathematical expressions to programming languages like FORTRAN. On NPOESS, an algorithm consists of: 1. A theoretical description (i.e., science/mathematical basis) 2. A computer implementation description (i.e., method of solution) 3. A computer implementation (i.e., code)
Algorithm Configuration Control Board (ACCB)	Interdisciplinary team of scientific and engineering personnel responsible for the approval and disposition of algorithm acceptance, verification, development and testing transitions. Chaired by the Algorithm Implementation Process Lead, members include representatives from IWPTB, Systems Engineering & Integration IPT, System Test IPT, and IDPS IPT.
Algorithm Verification	Science-grade software delivered by an algorithm provider is verified for compliance with data quality and timeliness requirements by Algorithm Team science personnel. This activity is nominally performed at the IWPTB facility. Delivered code is executed on compatible IWPTB computing platforms. Minor hosting modifications may be made to allow code execution. Optionally, verification may be performed at the Algorithm Provider's facility if warranted due to technical, schedule or cost considerations.
Ancillary Data	Any data which is not produced by the NPOESS System, but which is acquired from external providers and used by the NPOESS system in the production of NPOESS data products.
Auxiliary Data	Auxiliary Data is defined as data, other than data included in the sensor application packets, which is produced internally by the NPOESS system, and used to produce the NPOESS deliverable data products.
EDR Algorithm	Scientific description and corresponding software and test data necessary to produce one or more environmental data records. The scientific computational basis for the production of each data record is described in an ATBD. At a minimum, implemented software is science-grade and includes test data demonstrating data quality compliance.
Environmental Data Record (EDR)	*[IORD Definition]* Data record produced when an algorithm is used to convert Raw Data Records (RDRs) to geophysical parameters (including ancillary parameters, e.g., cloud clear radiation, etc.). *[Supplementary Definition]* An Environmental Data Record (EDR) represents the state of the environment, and the related information needed to access and understand the record. Specifically, it is a set of related data items that describe one or more related estimated environmental parameters over a limited time-space range. The parameters are located by time and Earth coordinates. EDRs may have been resampled if they are created from multiple data sources with different sampling patterns. An EDR is created from one or more NPOESS SDRs or EDRs, plus ancillary environmental data provided by others. EDR metadata contains references to its processing history, spatial and temporal coverage, and quality.
Operational Code	Verified science-grade software, delivered by an algorithm provider and verified by IWPTB, is developed into operational-grade code by the IDPS IPT.
Operational-Grade Software	Code that produces data records compliant with the System Specification requirements for data quality and IDPS timeliness and operational infrastructure. The software is modular relative to the IDPS infrastructure and compliant with IDPS application programming interfaces (APIs) as specified for TDR/SDR or EDR code.

Term	Description
Raw Data Record (RDR)	*[IORD Definition]* Full resolution digital sensor data, time referenced and earth located, with absolute radiometric and geometric calibration coefficients appended, but not applied, to the data. Aggregates (sums or weighted averages) of detector samples are considered to be full resolution data if the aggregation is normally performed to meet resolution and other requirements. Sensor data shall be unprocessed with the following exceptions: time delay and integration (TDI), detector array non-uniformity correction (i.e., offset and responsivity equalization), and data compression are allowed. Lossy data compression is allowed only if the total measurement error is dominated by error sources other than the data compression algorithm. All calibration data will be retained and communicated to the ground without lossy compression. *[Supplementary Definition]* A Raw Data Record (RDR) is a logical grouping of raw data output by a sensor, and related information needed to process the record into an SDR or TDR. Specifically, it is a set of unmodified raw data (mission and housekeeping) produced by a sensor suite, one sensor, or a reasonable subset of a sensor (e.g., channel or channel group), over a specified, limited time range. Along with the sensor data, the RDR includes auxiliary data from other portions of NPOESS (space or ground) needed to recreate the sensor measurement, to correct the measurement for known distortions, and to locate the measurement in time and space, through subsequent processing. Metadata is associated with the sensor and auxiliary data to permit its effective use.
Retrieval Algorithm	A science-based algorithm used to 'retrieve' a set of environmental/geophysical parameters (EDR) from calibrated and geolocated sensor data (SDR). Synonym for EDR processing.
Science Algorithm	The theoretical description and a corresponding software implementation needed to produce an NPP/NPOESS data product (TDR, SDR or EDR). The former is described in an ATBD. The latter is typically developed for a research setting and characterized as "science-grade".
Science Algorithm Provider	Organization responsible for development and/or delivery of TDR/SDR or EDR algorithms associated with a given sensor.
Science-Grade Software	Code that produces data records in accordance with the science algorithm data quality requirements. This code, typically, has no software requirements for implementation language, targeted operating system, modularity, input and output data format or any other design discipline or assumed infrastructure.
SDR/TDR Algorithm	Scientific description and corresponding software and test data necessary to produce a Temperature Data Record and/or Sensor Data Record given a sensor's Raw Data Record. The scientific computational basis for the production of each data record is described in an Algorithm Theoretical Basis Document (ATBD). At a minimum, implemented software is science-grade and includes test data demonstrating data quality compliance.
Sensor Data Record (SDR)	*[IORD Definition]* Data record produced when an algorithm is used to convert Raw Data Records (RDRs) to calibrated brightness temperatures with associated ephemeris data. The existence of the SDRs provides reversible data tracking back from the EDRs to the Raw data. *[Supplementary Definition]* A Sensor Data Record (SDR) is the recreated input to a sensor, and the related information needed to access and understand the record. Specifically, it is a set of incident flux estimates made by a sensor, over a limited time interval, with annotations that permit its effective use. The environmental flux estimates at the sensor aperture are corrected for sensor effects. The estimates are reported in physically meaningful units, usually in terms of an angular or spatial and temporal distribution at the sensor location, as a function of spectrum, polarization, or delay, and always at full resolution. When meaningful, the flux is also associated with the point on the Earth geoid from which it apparently originated. Also, when meaningful, the sensor flux is converted to an equivalent top-of-atmosphere (TOA) brightness. The associated metadata includes a record of the processing and sources from which the SDR was created, and other information needed to understand the data.

Term	Description
Temperature Data Record (TDR)	*[IORD Definition]* Temperature Data Records (TDRs) are geolocated, antenna temperatures with all relevant calibration data counts and ephemeris data to revert from T-sub-a into counts. *[Supplementary Definition]* A Temperature Data Record (TDR) is the brightness temperature value measured by a microwave sensor, and the related information needed to access and understand the record. Specifically, it is a set of the corrected radiometric measurements made by an imaging microwave sensor, over a limited time range, with annotation that permits its effective use. A TDR is a partially-processed variant of an SDR. Instead of reporting the estimated microwave flux from a specified direction, it reports the observed antenna brightness temperature in that direction.
Model Validation	The process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model. [Ref.: DoDD 5000.59-DoD Modeling and Simulation Management]
Model Verification	The process of determining that a model implementation accurately represents the developer's conceptual description and specifications. [Ref.: DoDD 5000.59-DoD Modeling and Simulation Management]

3.2 Acronyms

The current acronym list for the NPOESS program, D35838_G_NPOESS_Acronyms, can be found on eRooms. Table 43 contains those terms most applicable for this OAD.

Table 43. Acronyms

Term	Description
None	

4.0 OPEN ISSUES

Table 44. TBXs

TBX ID	Title/Description	Resolution Date
None		