

# NWP SAF

## *Satellite Application Facility for Numerical Weather Prediction*

Document NWPSAF-MF-UD-002

Version 8.1, March 2019

## **AAPP DOCUMENTATION SOFTWARE DESCRIPTION**

AUTHORS : Tiphaine Labrot (Météo-France)  
Nigel Atkinson (Met Office)  
Pascale Roquet (Météo-France)



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

This documentation was developed within the context of the EUMETSAT Satellite Application Facility on Numerical Weather Prediction (NWP SAF), under the Cooperation Agreement dated 01 December 2006, between EUMETSAT and the Met Office, UK, by one or more partners within the NWP SAF. The partners in the NWP SAF are the Met Office, ECMWF, KNMI and Météo France.

Copyright 2017, EUMETSAT, All Rights Reserved.

Change record			
Version	Date	Author / changed by	Remarks
4.0	April 03	T.Labrot	Version1 of the software description document of AAPP V4.0 (Follow the version of AAPP V3.0 )
4.1	12 May 2003	K Whyte	Minor edit
5.0	March 2005	T.Labrot N C Atkinson P. Brunel	Update for AAPP V5
6.0	June 2006	T.Labrot N C Atkinson	Update for AAPP V6
7.0	Jan 2012	T.Labrot N C Atkinson	Update for AAPP V7
7.1	July 2012	P Roquet N C Atkinson	Insert sections on MMAM and modify atovin/atovpp descriptions, for release of AAPP v7.2.
7.2	Feb 2013	P Roquet N C Atkinson	Add sections on MAIA4
7.3	Feb 2014	P Roquet N C Atkinson	Add sections on NOAA/CLASS conversion tools and update MAIA4 section
7.4	Aug 2014	N C Atkinson	Updates for MWTS2, MWHS2 and IRAS
7.5	Dec 2014	P Roquet	Update for MAIA v4.2 release.
7.6	May 2015	N C Atkinson	Update section on hrptdc and add viirs_to_cris.
8.0	Dec 2017	P Roquet N C Atkinson	Update for AAPP V8
8.1	March 2019	N C Atkinson	Add descriptions of VIIRS_CRIS_MAIA4_RUN and aapp1c_quicklook.py

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

## TABLE OF CONTENTS

<b>1. INTRODUCTION .....</b>	<b>8</b>
<b>2. DOCUMENTS AND TERMINOLOGY .....</b>	<b>8</b>
2.1. APPLICABLE AND REFERENCE DOCUMENTS .....	8
2.2. TERMINOLOGY .....	10
<b>3. SOFTWARE ORGANISATION DESCRIPTION .....</b>	<b>12</b>
3.1. SOFTWARE GENERAL ORGANISATION .....	12
3.1.1. <i>The core AAPP</i> .....	12
3.1.2. <i>METOP tools</i> .....	13
3.1.3. <i>IASI tools</i> .....	14
3.2. INTERFACES .....	14
3.3. DIAGRAMS .....	15
3.4. DIRECT-READOUT OF NOAA SATELLITE DATA .....	15
3.5. DIRECT-READOUT OF METOP SATELLITE DATA .....	18
3.6. ACQUISITION OF METOP DATA VIA EUMETCAST .....	23
3.6.1. <i>NOAA archived data</i> .....	25
<b>4. GENERAL DESCRIPTION .....</b>	<b>25</b>
4.1. SOFTWARE MAIN COMPONENTS .....	25
4.1.1. <i>Main module for direct-readout of NOAA satellites. AAPP_RUN_NOAA script</i> .....	25
4.1.2. <i>Main module for direct-readout of MetOp satellite. AAPP_RUN_METOP script</i> .....	26
4.1.3. <i>Main module for FY1 imager data. AAPP_RUN_FY1 script</i> .....	27
4.1.4. <i>Satellite and image navigation initialisation: Ingest with TBUS bulletin, TBUSING script, TBUSING.EXE and satellite position and velocity: SATPOST script, SATPOST.EXE</i> .....	27
4.1.5. <i>Satellite and image navigation initialization with Two Line Element sets: GET_TLE script, TLEING script, TLEING.EXE and satellite position and velocity: SATPOSTLE script, SATPOSTLE.EXE</i> .....	30
4.1.6. <i>Decommutation modules: DECOMMUTATION script and DECOMMUTATION.EXE</i> .....	34
4.1.7. <i>EPS level 0 to AAPP level 1a conversion for METOP: DECOM-AMSUA-METOP script and AMSUA-MAIN.EXE, DECOM-MHS-METOP script and MHS-MAIN.EXE, DECOM-AMSUA-HIRS script and HIRS-MAIN.EXE, DECOM-AVHRR-METOP script and AVHRR-MAIN.EXE</i> .....	39
4.1.8. <i>Convert chrpt (FY1c and FY1d satellites) to hrpt (NOAA satellites): convert_chrpt script and convert_chrpt.exe</i> .....	40
4.1.9. <i>Image navigation modules: HIRSCL script and HIRSCL.EXE, HIRSCL_ALGOV4 script and HIRSCL_ALGOV4.EXE, MSUCL script and MSUCL.exe, AMSUACL script and AMSUACL.EXE, AMSUBCL script and AMSUBCL.EXE, MHSCL script and MHSCL.EXE, AVHRCL script and AVHRCL.EXE</i> .....	41
4.1.10. <i>HIRS calibration modules (first algorithm): HIRSCL script and HIRSCL.EXE</i> .....	43
4.1.11. <i>HIRS calibration modules (algorithm version 4): HCALCB1_ALGOV4 script and HCALCB1_ALGOV4.EXE, HIRSCL_ALGOV4 script and HIRSCL_ALGOV4.EXE</i> .....	46
4.1.12. <i>MSU calibration modules: MSUCL script and MSUCL.EXE</i> .....	49
4.1.13. <i>AMSU-A calibration modules : AMSUACL script and AMSUACL.EXE</i> .....	52
4.1.14. <i>AMSU-B calibration modules: AMSUBCL script and AMSUBCL.EXE</i> .....	55
4.1.15. <i>MHS calibration modules: MHSCL script and MHSCL.EXE</i> .....	57
4.1.16. <i>AVHRR calibration module: AVHRCL script and AVHRCL.EXE</i> .....	60
4.1.17. <i>ATOVS sounders calibration: ATOVIN script and ATOVIN.EXE</i> .....	61
4.1.18. <i>Mapping of sounders: ATOVPP script and ATOVPP.EXE</i> .....	64

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

4.1.19. Modify the ATMS beam width: ATMS_BEAMWIDTH script, ATMS_BEAMWIDTH.EXE	82
4.1.20. Modify the MWTS2 or MWHS2 beam width: MWTS2_BEAMWIDTH and MWHS2_BEAMWIDTH scripts, MWTS2_BEAMWIDTH.EXE and MWHS2_BEAMWIDTH.EXE	82
4.1.21. Mapping AVHRR to HIRS + Cloud Mask: AVH2HIRS script , AVH2HIRS.EXE or AVH2HIRS_ATOVS.EXE.	82
4.1.22. AVHRR calibration: AVHRRIN script and AVHRRIN.EXE	87
4.1.23. Convert AVHRR AAPP 11b format to AVHRR PFS LIB format: AAPP-EPS AVHRRLIB script and EPS_AVHRRLIB-MAIN.EXE.	89
4.1.24. Convert IASI PFS LIC to IASI AAPP 11c : CONVERT_IASIIIC, CONVERT_IASIIIC.EXE and CONVERT_IASIIIC_9.0.EXE.	90
4.1.25. Convert NOAA 11b formats to AAPP 11b format: noaa_class_to_aapp script and associated executables.	90
4.1.26. Convert AVHRR 11b in AAPP format to NOAA format: avhrr_aapp_to_class script and avhrr_aapp_to_class.exe	91
4.1.27. Initialisation before OPS-LRS software: SATPOS-SVM.KSH, SATPOS-SVM.PL	91
4.1.28. Initialisation before OPS-LRS software: MESSAGES-OSV.KSH, MESSAGES-OSV.PL	91
4.1.29. Navigation tools: SATEPH script, LGEPHEING script and LGEPHING.EXE, LGEPHE script and LGEPHE.EXE, ALLEPH script and EPHE, TRACKING, ANTCNFT, DRIFTEPHE, TBUSDISP script, TBUSDISP.EXE, TLEPRINT script, TLEPRINT.EXE.	91
4.1.30. BUFR tools	96
4.1.31. HDF5 tools	98
4.2. INTERFACES	100
4.2.1. User input parameters in ATOVS_ENV/ATOVS_ENV8	100
4.2.2. Inputs/outputs for TBUSING navigation initialisation	101
4.2.3. Inputs/outputs for GET_TLE navigation initialization	101
4.2.4. Inputs/outputs for GET_TAI_UTI.UTC navigation tool	102
4.2.5. Inputs/outputs for TLEING navigation initialisation	102
4.2.6. Inputs/outputs for SATPOST navigation initialisation	103
4.2.7. Inputs/outputs for SATPOSTLE navigation initialisation	104
4.2.8. Inputs/outputs for decommutation (DECOMMUTATION)	104
4.2.9. Inputs/outputs EPS level 0 format to AAPP level 1a format	106
4.2.10. Inputs/outputs ATOVS and AVHRR navigation (HIRSCL, HIRSCL_ALGOV4, MSUCL, AMSUACL, AMSUBCL, MHSCL, AVHRCL)	107
4.2.11. Inputs/outputs HIRS calibration (first algorithm) (HIRSCL)	108
4.2.12. Inputs/outputs HIRS calibration algorithm version 4 – part 1 (HCALCB1_ALGOV4)	109
4.2.13. Inputs/outputs HIRS calibration algorithm version 4 – part 2 (HIRSCL_ALGOV4)	110
4.2.14. Inputs/outputs MSU calibration (MSUCL)	111
4.2.15. Inputs/outputs AMSU-A calibration (AMSUACL)	112
4.2.16. Inputs/outputs AMSU-B calibration (AMSUBCL)	114
4.2.17. Inputs/outputs MHS calibration (MHSCL)	115
4.2.18. Inputs/outputs AVHRR calibration (AVHRCL)	117
4.2.19. Inputs/outputs sounders calibration application (ATOVIN)	118
4.2.20. Inputs/outputs sounders mapping(ATOVPP)	120
4.2.21. Inputs/outputs for mapping cloud mask AVHRR to HIRS (AVH2HIRS)	123
4.2.22. Inputs/outputs sounders calibration application (AVHRRIN)	124
4.2.23. Inputs/outputs for conversion AVHRR AAPP 11b format to AVHRR PFS LIB format (aapp-eps_avhrr11b)	125
4.2.24. Inputs/outputs for SATEPH navigation tool	126

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

4.2.25. Inputs/outputs for LGEPHEING navigation tool.....	126
4.2.26. Inputs/outputs for LGEPHE navigation tool.....	127
4.2.27. Inputs/outputs for ALLEPH navigation tool.....	128
4.2.28. Inputs/outputs for TBUSDISP navigation tool.....	129
4.2.29. Inputs/outputs for TLEPRINT navigation tool.....	129
4.2.30. Inputs/outputs for EPHE, TRACKING, ANTCNFT, DRIFTEPHE navigation tool.....	130
4.3. DYNAMIC ARTICULATION.....	130
4.3.1. Description of the main script AAPP_RUN_NOAA.....	130
4.3.2. Description of the script CHK1BTIME.....	131
4.3.3. Description of the script TBUSING.....	131
4.3.4. Description of the script GET_TLE.....	131
4.3.5. Description of the script TLEING.....	131
4.3.6. Description of the script SATPOST.....	132
4.3.7. Description of the script SATPOSTLE.....	132
4.3.8. Description of the script DECOMMUTATION.....	132
4.3.9. Description of the scripts HIRSCL, HIRSCL_ALGOV4, MSUCL, AMSUCL, AMSUBCL, MHSCL, AVHRCL.....	134
4.3.10. Description of the script ATOVIN.....	134
4.3.11. Description of the script ATOVPP.....	135
4.3.12. Description of the script AVH2HIRS.....	135
4.3.13. Description of the script AVHRRIN.KSH.....	136
4.3.14. Description of the script EPS_AVHRRLIB-MAIN.....	136
4.3.15. Description of the script EPS_CONVERT_IASL1C.....	136
4.3.16. Description of the script NOAA_CLASS_TO_AAPP.....	136
4.3.17. Description of the script AVHRR_AAPP_TO_CLASS.....	137
4.3.18. Description of the script SATPOS-SVM.....	137
4.3.19. Description of the script MESSAGES-OSV.....	137
4.3.20. Description of the script SATEPH.....	137
4.3.21. Description of the script LGEPHEING.....	137
4.3.22. Description of the script LGEPHE.....	138
4.3.23. Description of the script ALLEPH.....	138
4.3.24. Description of the command EPHE.....	138
4.3.25. Description of the command TRACKING.....	139
4.3.26. Description of the command ANTCNFT.....	139
4.3.27. Description of the command DRIFTEPHE.....	139
4.3.28. Description of the script TBUSDISP.....	139
4.3.29. Description of the script TLEPRINT.....	139
4.3.30. AVHRR and HIRS level 1b file verification : PRHAVH and PRHIRS.....	140
4.3.31. MSU level 1b file header verification PRHMSU.....	140
4.3.32. DCS level 1b file verification PRHDCS.....	140
4.3.33. Source file identification: HRPTIDF.....	140
4.3.34. Level 1b products identification: LIBIDF.....	141
4.3.35. Level 1c products identification: LICIDF.....	141
4.3.36. Level 1d products identification: LIDIDF.....	141
4.3.37. Write out a message: LIBLOG.....	141
4.3.38. Get the orbit number: SDH2ORBNUM.....	142
4.3.39. Decode 1c BUFR files: AAPP_DECODEBUFR_1C or ECCODES_DECODEBUFR_1C .....	142
4.3.40. Encode 1c BUFR files : AAPP_ENCODEBUFR_1C or ECCODES_ENCODEBUFR_1C .....	142

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

4.3.41. Decode Sensor Data Record files for ATMS, CrIS, MWTS, MWHS, MWTS2, MWHS2, IRAS, HIRAS.....	143
4.3.42. FY-3 mapping tools: mwhs_to_mwts, mwhs2_to_mwts2, mwts2_to_mwts2, mwts2_to_iras, mwhs2_to_iras.....	143
4.3.43. is-mmam .exe.....	144
4.3.44. mmam-main .exe.....	144
4.3.45. print-mmam-obt-utc.pl.....	144
4.3.46. patch-level0-from-mmam.exe.....	145
4.3.47. atms1c_print_nedt.....	145
4.3.48. aapp1c_quicklook.py.....	145
4.4. MAIA.....	145
4.4.1. Introduction.....	145
4.4.2. MAIA interfaces.....	147
4.4.3. Usage.....	148
4.4.4. USER INPUT PARAMETERS FOR MAIA4 IN ATOVS_ENV.....	149
4.4.5. MAIA Software components.....	150
4.4.6. MAIA file source codes description.....	152
4.5. MAIA2PFS.EXE.....	153
4.6. VIIRS TOOLS.....	153
4.6.1. Decode and concatenate Sensor Data Record granule files for VIIRS.....	153
4.6.2. Decode EDR IMG granule files for VIIRS.....	154
4.6.3. The Fortran90 aapp_viirs API.....	154
4.6.4. VIIRS to CrIS mapping.....	159
4.6.5. VIIRS_CRIS_MAIA4_RUN.....	160

## Figures

FIGURE 3-1 : FIRST STEPS FOR TREATING NOAA DATA.....	16
FIGURE 3-2 : SECOND STEPS FOR TREATING NOAA DATA.....	17
FIGURE 3-3 : PRE-PROCESSING STEPS FOR NOAA DATA.....	18
FIGURE 3-4 : PERIODICAL STEP FOR TREATING METOP DATA.....	19
FIGURE 3-5 : FIRST STEPS FOR TREATING METOP DATA (ATOVS PART).....	19
FIGURE 3-6 : SECOND STEPS FOR TREATING METOP DATA (ATOVS PART).....	20
FIGURE 3-7 : FIRST STEPS FOR TREATING METOP DATA (IASI PART).....	21
FIGURE 3-8 : PRE-PROCESSING STEPS FOR METOP DATA.....	22
FIGURE 3-9 : CHAIN FOR TREATING METOP -ATOVS DATA RECEIVED VIA EUMETCAST.....	23
FIGURE 3-10 : CHAIN FOR TREATING METOP –AVHRR - HIRS DATA RECEIVED VIA EUMETCAST.....	24
FIGURE 3-11 : CHAIN FOR TREATING ARCHIVED NOAA DATA.....	25
FIGURE 4-1 : FLOW CHART ON THE COMPONENTS OF THE TBUSING MODULE.....	28
FIGURE 4-2 : FLOW CHART ON THE SATPOST MODULE COMPONENTS.....	29
FIGURE 4-3 : FLOW CHART ON THE COMPONENTS OF THE TLEING MODULE.....	31
FIGURE 4-4 : FLOW CHART ON THE SATPOSTLE MODULE COMPONENTS.....	33
FIGURE 4-5 : DECOMMUTATION AND HRPTDC MODULE HIERARCHY.....	35
FIGURE 4-6 : ATOVDC COMPONENTS HIERARCHY.....	36
FIGURE 4-7 : AVHRDC COMPONENTS HIERARCHY.....	37
FIGURE 4-8 : GENERAL FLOW CHART ON THE LOCATION MODULE COMPONENTS : HIRSCL/MSUCL/AMSUNCL/MHSCL/AVHRCL.....	41

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

FIGURE 4-9 : FLOW CHART ON THE HIRSCL MODULE COMPONENTS..... 44

FIGURE 4-10 : FLOW CHART ON THE HIRSCL\_ALGOV4 MODULE COMPONENTS..... 47

FIGURE 4-11 : FLOW CHART ON THE MSUCL MODULE COMPONENTS..... 50

FIGURE 4-12 : FLOW CHART ON THE AMSUCL MODULE COMPONENTS..... 53

FIGURE 4-13 : FLOW CHART ON THE AMSUBCL MODULE COMPONENTS..... 56

FIGURE 4-14 : FLOW CHART ON THE AMSUBCL AND MHSCL MODULE COMPONENTS..... 58

FIGURE 4-15 : FLOW CHART ON THE AVHRCL MODULE COMPONENTS..... 61

FIGURE 4-16 : ATOVIN MODULE HIERARCHY ..... 62

FIGURE 4-17 : INAMSA MODULE HIERARCHY ..... 63

FIGURE 4-18 : ATOVPP MODULES HIERARCHY ..... 65

FIGURE 4-19 : PPSETUP MODULES HIERARCHY ..... 68

FIGURE 4-20 : PPLUT MODULES HIERARCHY ..... 69

FIGURE 4-21 : PPIN MODULES HIERARCHY ..... 70

FIGURE 4-22 : PPPROC1 MODULES HIERARCHY..... 74

FIGURE 4-23 : PPMAP MODULES HIERARCHY..... 76

FIGURE 4-24 : PPPROC2 MODULES HIERARCHY..... 78

FIGURE 4-25 : PPOUT MODULES HIERARCHY..... 80

FIGURE 4-26 : PPFINISH MODULES HIERARCHY ..... 81

FIGURE 4-27 : AVH2HIRS\_ATOVS/AVH2HIRS MODULES HIERARCHY ..... 83

FIGURE 4-28 : AVHRRIN MODULES HIERARCHY ..... 87

FIGURE 4-29 : MAIA INTERFACES ..... 147

FIGURE 4-30 : MAIA COMPONENTS FOR AVHRR..... 151

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

## **1. INTRODUCTION**

For many years the NOAA polar orbiting weather satellites have provided a sounding and imaging capability, with instruments operating in the visible, infra-red and microwave regions of the spectrum, and with a direct broadcast system to allow users access to the data in near real time.

In response to requests from the user community, EUMETSAT took the initiative in 1992 to start activities in the area of ATOVS software processing. The goal was to set up a standard package for the processing of locally received ATOVS data from the NOAA spacecraft, and as a result of this initiative the ATOVS and AVHRR Pre-processing Package (AAPP) was developed. The package is now maintained by the EUMETSAT Satellite Application Facility for Numerical Weather Prediction (NWP SAF).

The first satellite in the NOAA-KLM series (NOAA-15) was launched in 1998, replacing the earlier NOAA/TIROS-N series. In 2009, the last satellite in the follow-on NOAA-NN' series was launched (NOAA-19), and the AAPP package (versions 5 and 6) was extended to accept data from this series.

A next major development was the launch in 2006 of the first European METOP satellite. METOP is part of the EUMETSAT Polar System (EPS), which is the European contribution to a joint European-US polar satellite system called the Initial Joint Polar System (IJPS). METOP capability was added in AAPP v6. The ability to process imager data from the Chinese FY-1D satellite was also added as part of AAPP v6.

The first of the next generation of US operational polar-orbiting weather satellites is the NPP (NPOESS Preparatory Project), launched in October 2011. Future satellites in the series will be named JPSS (Joint Polar Satellite System). AAPP v7 was designed to pre-process data from the sounder and imager instruments on NPP, while continuing to support MetOp and the older NOAA satellites.

AAPP v8 includes updates to the MAIA cloud mask software, and to the BUFR encode/decode software.

This document provides a software description of the AAPP package. It includes a description of the software modules for processing ATOVS and AVHRR data on METOP, but excludes the IASI level 0 to level 1c convertor, OPS-LRS, which is described in the OPS-LRS User Manual.

## **2. DOCUMENTS AND TERMINOLOGY**

### **2.1. APPLICABLE AND REFERENCE DOCUMENTS**

[1]: NESS 107: 'Data Extraction and Calibration of TIROS-N/NOAA Radiometer'. NOAA Technical Memorandum - Planet, 1988.

And the NOAA KLM user's guide available from  
<https://www1.ncdc.noaa.gov/pub/data/satellite/publications/podguides/>

[2]: 'General specifications for the AAPP preprocessing package related to NOAA polar orbiting weather satellites. Scientific part'. Météo France internal document- 1999.

[3]: 'General specifications for the AAPP preprocessing package related to NOAA polar orbiting weather satellites. Software description'. Météo France internal document - 1999.

[4]: 'AAPP Module Design' - 'AAPP Data Set Definition'. Documentation EUMETSAT - Vol1 and Vol2 - 1997.

[5]: 'Measurement of the AMSU-B Antenna Pattern'. T.J. Hewison & R. Saunders, IEEE Transactions of Geosciences and Remote Sensing, Vol. 34 No 2, Mars 1996.

[6]: 'Estimating the probability of rain in an SSM/I FOV using logistic regression'. Crosby, Ferraro & Wu, Journal of Applied Met., Vol 34 No 11, 1995.



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

- [7]: Deleted.
- [8]: Deleted
- [9]: Deleted
- [10]: Grody N. ‘Classification of snow cover and precipitation using the Special Sensor Microwave Imager’. J. Geophys. Res., vol 96, 199.
- [11]: Gutman G., D. Tarpley, A. Ignatov, S. Olson, The enhanced NOAA global dataset from the advanced very high resolution radiometer. Bulletin of the American Meteorological Society. 1995.
- [12]: Lavanant L., H. LeGléau, M. Derrien, S. Levasseur, G. Monnier, L. Ardouin, P. Brunel, B. Bellec: AVHRR Cloud Mask for Sounding Applications. ITSC-10 proceedings, 1999.
- [13]: Oort A.: Global Atmospheric Circulation Statistics. 1958 –1973.
- [14]: Saunders R.: ‘An automated scheme for the removal of cloud contamination from AVHRR radiances over western Europe’. Int. J. Remote sensing, 1986..
- [15]: Saunders R.: ‘An improved method for detecting clear sky and cloudy radiances from AVHRR data’. Int. J. Remote Sensing, 1988..
- [16]: Deleted
- [17]: Brunel P. and Marsouin A., 2000, Operational AVHRR navigation results, *International Journal of Remote Sensing*, Vol. 21, No. 5, 951-972.
- [18]: Rosborough G.W., Baldwin D. and Emery W., 1994, Precise AVHRR Image Navigation, *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 32, No. 3, May 1994, 644-657.
- [19]: Level 1B Notices, <http://www.osdpd.noaa.gov/ml/ppp/notices.html>
- [20]: Brunel P. and Marsouin A., 2001, ANA-3 User’s Manual, Meteo-France/DP/Centre de Meteorologie Spatiale, BP 147, 22302 Lannion, France.
- [21]: Bordes Ph., Brunel P. and Marsouin A., 1992, Automatic Adjustment of AVHRR Navigation, *Journal of Atmospheric and Oceanic Technology*, Vol. 9, No. 1, February 92.
- [22]: Marsouin A., Brunel P., AAPP Documentation, Annex of scientific description, AAPP navigation, document NWPSAF-MF-UD-005, distributed with AAPP
- [23]: Changyong Cao NESDIS, HIRS Calibration Algorithm Version 4.0
- [24]: Changyong Cao and Pubu Ciren, Operational High Resolution Infrared Radiation Sounder (HIRS) Calibration Algorithms and Their Effects on Calibration Accuracy, ITSC XIII Proceedings (2003), [cimss.ssec.wisc.edu/itwg/itsc/itsc13/session3/3\\_2\\_ciren.pdf](http://cimss.ssec.wisc.edu/itwg/itsc/itsc13/session3/3_2_ciren.pdf)
- [25]: Bennartz, Thoss, Dybbroe and Michelson, ‘Precipitation analysis using the Advanced Microwave Sounding Unit in support of nowcasting applications’, *Meteorol. Appl.*, 9, 177-189, 2002
- [26]: Lee, A.C.L. and Bedford, S., ‘Support Study on IASI Level 1c Data Compression’, Final Report, EUMETSAT Contract EUM/CO/-3/1155/PS, Feb 27, 2004
- [27]: Goldberg et. al., “AIRS Near-Real-Time Products and Algorithms in Support of Operational Numerical Weather Prediction”, *IEE Trans. Geosci. Rem. Sens.*, vol. 41, no. 2, Feb 2003.
- [28]: Collard, A.D., “Selection of IASI channels for use in numerical weather prediction”, ECMWF Technical Memorandum 532, July 2007.
- [29]: “OPS-LRS User Manual”, document NWPSAF-MF-UD-006, distributed with AAPP.
- [30]: “AAPP Version 7 Top Level Design”, document NWPSAF-MO-DS-011, distributed with AAPP.
- [31]: “EPS Programme Generic Product Format Specification”, document EPS-GGS-SPE-96167, available from [www.eumetsat.int](http://www.eumetsat.int)
- [32]: “EPS/MetOp Technical Note on Orbit Prediction” Conzalo Garcia-Julian, Miguel M. Romany Merino - GMSA SA 1997

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

[33]: “Annex to AAPP scientific documentation: Pre-processing of ATMS and CrIS”, document NWPSAF-MO-UD-027, distributed with AAPP.

[34]: “IASI Principal Components in AAPP: User Manual”, document NWPSAF-MO-UD-022, distributed with AAPP.

[35]: “MAIA AVHRR Cloud Mask and Classification”, L. Lavanant, document MF/DP/CMS/R&D/MAIA3, 2002, available at [www.meteorologie.eu.org/ici/maia/maia3.pdf](http://www.meteorologie.eu.org/ici/maia/maia3.pdf)

[36]: “JPSS Common Data Format Control Book” volumes I to VIII, available at <https://jointmission.gsfc.nasa.gov/documents.html>

[37]: “Annex to AAPP scientific documentation: Pre-processing of ATMS and CrIS”, document NWPSAF-MO-UD-027

[38]: “VIIRS-CrIS mapping”, document NWPSAF-MF-UD-011

[39]: “MAIA version 4 for NPP-VIIRS and NOAA/METOP-AVHRR cloud mask and classification. Scientific user manual”, document NWPSAF-MF-UD-009

## 2.2. TERMINOLOGY

**AAPP:** ATOVS and AVHRR Pre-processing Package.

**ADC:** Analog to Digital Converter.

**AIP:** AMSU Information Processor.

**AMSU:** Advanced Microwave Sounding Unit.

**ANA:** Automatic Navigation Adjustment.

**ARGOS:** Name of the orbital bulletin emitted by CLS/ARGOS.

**Ascending node (HNA) :** equator satellite crossing when it comes from south pole.

**ATMS:** Advanced Technology Microwave Sounder

**ATOVS:** Advanced TIROS Vertical Sounder.

**AVHRR:** Advanced Very High Resolution Radiometer.

**Attitude:** Satellite orientation according the 3 axes (yaw, roll, pitch).

**Bb:** black body.

**Brolzyd (Brouver-Lyddane):** Orbit extrapolation model algorithm for TBUS bulletin.

**CMS:** Centre de Météorologie Spatiale (Météo-France)

**CNES:** Centre National d'études Spatiales.

**CrIS :** Cross-track Infrared Sounder

**DCS:** Data Collection System.

**Descending node (LNA) :** equator satellite crossing when it comes from north pole.

**DMSP:** Defense Meteorological Satellite Program

**DWSS:** Defense Weather Satellite System

**Earth's precession:** Slow conical motion of the Earth rotation axis around a mean position corresponding to a normal direction to the ecliptic plane.

**Ecliptic plane:** The Earth orbital plane around the Sun.

**ECMWF:** European Center for Medium Weather Forecasting.

**Ephemeris:** The list of the times of various events as: ascending and descending nodes, start and end of acquisition by a station.

**EPS:** EUMETSAT Polar System

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**FOV:** Field Of View.

**GAC:** Global Area Coverage.

**HIRS:** High Resolution Infra Red Sounder.

**HRPT:** High Resolution Picture Transmission.

**IASI:** Infrared Atmospheric Sounding Interferometer.

**IFOV:** Instantaneous Field Of View.

**IJPS:** Initial Joint Polar System

**Image navigation:** Conversion of line and pixel numbers into latitude and longitude.

**IR:** InfraRed.

**IWT :** internal warm target

**LAC:** Local Area Coverage.

**Mapping :** for sounders = computing sounder data to another sounder grid. For imaging radiometer = imaging radiometer data segmentation to sounder ellipse.

**MetOp:** Meteorological Operational satellite

**MHS:** Microwave Humidity Sounder

**MIRP:** Manipulated Information Rate Processor.

**MSU:** Microwave Sounding Unit.

**μ-waves:** microwaves.

**Nadir:** Satellite vertical direction.

**NESDIS:** National Environmental Satellite Data Information Service.

**NOAA:** National Oceanic and Atmospheric Administration.

**NORAD:** North American Aerospace Defense Command

**NPP:** NPOESS Preparatory Project

**NWP SAF :** Numerical Weather Prediction Satellite Application Facility.

**Perigee:** Satellite orbit point which is the nearest from the Earth (opposite apogee).

**PM:** Pulse Modulation.

**POES:** Polar Orbiting Environmental Satellite(s)

**PRT:** Platinum Resistance Thermometer.

**Rg :** Greenwich reference frame

**RI :** local reference frame

**Rs :** spacecraft fixed reference frame

**Rv :** satellite local orbital frame

**SDP4:** Orbit extrapolation model for deep-space object Two-Line Element sets

**SEM:** Space Environment Monitor.

**SGP4:** Orbit extrapolation model for near-Earth object Two-Line Element sets

**SSU:** Stratospheric Sounding Unit.

**SST:** Sea Surface Temperature.

**TBUS:** Name of the orbital bulletin published by NOAA/NESDIS.

**TIP:** TIROS Information Processor.

**TIROS:** Television Infrared Observation Satellite

**TLE:** Two-Line elements, name of the orbital bulletin published by NORAD or satellite agencies

**TOVS:** TIROS Operational Vertical Sounder.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**VIIRS:** Visible/Infrared Imager/Radiometer Suite

**VIS:** Visible.

### **3. SOFTWARE ORGANISATION DESCRIPTION**

#### **3.1. SOFTWARE GENERAL ORGANISATION**

AAPP version 8 presents three distinct components:

The core AAPP task.

Tools to interface the core AAPP with the specific formats of METOP data (located under the directory metop-tools).

A suite for processing IASI data to level 1c, based on the CNES-supplied IASI OPS (Operational Software), named OPS-LRS for Local Reception Station. The OPS-LRS package has its own self-contained directory structure, but to run it requires the use of a set of tools containing format libraries, conversion tools, etc. (located under the directory iasi-tools).

##### **3.1.1. The core AAPP**

The core AAPP can be broken down into seven major tasks:

Ingest step 1: Decommuration (only useful for direct acquisition of NOAA satellite data)

Ingest step 2: Calculation of calibration coefficients/satellite navigation/localisation

Preprocessing step 1 (atovin): Main function: Apply calibration coefficients, convert radiances to brightness temperatures.

Preprocessing step 2 (atovpp): Main function: Instrument mapping on another instrument grid.

Preprocessing step 3 (avh2hirs): AVHRR mapping on HIRS and cloud mask. This step is only available for HIRS, as the name shows.

A cloud mask at the full resolution of the AVHRR (maia4)

Tools to perform a range of tasks, including BUFR encode/decode, reading/writing of HDF5 files, etc.

##### ***Ingest***

---

##### **DECOMMUTATION:**

**DECOMMUTATION** performs the interface between acquisition system and processing. This function is specific to the AAPP installation site and can be modified by the user if the acquisition system doesn't respect HRPT format. This module calls **HRPTDC** to perform decommutation task.

**HRPTDC** reads the raw (level 0) HRPT data streams and puts data from the sounding instruments (HIRS, AMSU-A, AMSU-B, MHS, MSU) and from the AVHRR radiometer into separated files (level 1a).

##### **SATELLITE AND IMAGES NAVIGATION - CALIBRATION COEFFICIENTS:**

**HIRSCL** or **HIRSCL\_ALGOV4**, **AMSUACL**, **AMSUBCL**, **MHSCL**, **MSUCL** perform the satellite navigation, the Earth localisation of the pixels, and the calibration coefficients calculation

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

for each TOVS/ATOVS instrument. Two algorithms are available to calibrate the HIRS, the user has to choose between HIRSCL or HIRSCL\_ALGOV4 at the AAPP installation.

AVHRCL performs the same tasks for the AVHRR radiometer.

At the end of this step, separated files of Earth located and calibration coefficients exist. Those (level 1b) files are archived.

### Pre-Processing:

---

#### **CALIBRATION:**

ATOVIN applies the calibration coefficients calculated by the previous step (**HIRSCL or HIRSCL\_ALGOV4, AMSUACL, AMSUBCL, MHSCCL, MSUCL**) to the numeric counts for radiance conversion. Before, for AMSU-A data a moon detection/correction is done and for the AMSU-B bias corrections and antenna corrections are added. Then **ATOVIN** converts each channel radiance into brightness temperature for each TOVS/ATOVS instrument. At the end of this procedure, separated files of Earth located brightness temperature data exist. Those (level 1c) files are archived.

#### **MAPPING:**

ATOVPP recognises the data contaminated by precipitation and maps data between the measurement grids of the different instruments (for example: HIRS + AMSU-A + AMSU-B on HIRS grid, HIRS+MSU on HIRS grid, AMSU-A + AMSU-B on AMSU-B grid).

#### **MAPPING - CLOUD MASK:**

AVH2HIRS applies the calibration coefficients (calculated by **AVHRCL**) to AVHRR counts and converts radiance into brightness temperature, maps AVHRR data in HIRS FOV, and (optionally) makes the cloud mask in the HIRS ellipse for contaminated pixels discrimination. At the end of this procedure, a level 1d file exists (HIRS level 1d).

### MAIA4:

---

#### **CALIBRATION:**

AVHRRIN applies the calibration coefficients calculated by the previous step (**AVHRCL**) to AVHRR counts and converts radiance into brightness temperature (avhrr.11c file).

#### **CLOUD MASK:**

MAIA4\_AVHRR makes the cloud mask at full resolution for the AVHRR imager (avhCT.h5 file).

MAIA4 makes the cloud mask at full resolution for the VIIRS imager (viiCT.h5 file)

Specific libraries are associated at all this main modules.  
 Each module is described in more detail in the section 3.2.  
 MAIA modules are described in section 4.4.

### **3.1.2. METOP tools**

To process the METOP data, a set of tools have been developed to interface the PFS level 0 format to the AAPP level 1a/1b format: One script/one main program by instrument: **DECOM-HIRS-METOP/HIRS-MAIN.EXE, DECOM-AMSUA-METOP/AMSUA-MAIN.EXE, DECOM-MHS-METOP/MHS-MAIN.EXE, DECOM-AVHRR-METOP/AVHRR-MAIN.EXE.**

Another tool (**AAPP-EPS\_AVHRR11B/EPS\_AVHRR11B-MAIN.EXE**) interfaces the AVHRR AAPP level 1b format to the AVHRR PFS level 1B format. The PFS resulting file has only partial

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

contents and is primarily intended for use in IASI OPS-LRS processing. The AVHRR PFS level 1B format is used by EUMETSAT for distribution of global AVHRR data, therefore a tool **convert\_avh1b** can be used to convert back to AAPP level 1b format (but with scaled radiances instead of raw counts).

### 3.1.3. IASI tools

Several modules and C libraries have been developed to handle the data related to the IASI OPS-LRS. OPS-LRS needs several files as input:

an OBT file that includes the difference between the atomic time and the UTC time. The modules `eps_metopl0-obt-xml.ksh/eps_metopl0-obt-xml.c` create this file from the IASI PFS L0.

an OSV file that contains data related to satellite manoeuvres. `messages-osv.ksh/messages-osv.pl` create this file from the ADMIN message.

an SVM file that includes the start and the end of the shadow. `satpos-svm.ksh/satpos-svm.pl` create this file from the satpos file.

The following modules are used to switch delivered files from big-endian to little endian:

`cnes_iasi_brd-swapb.ksh/cnes_iasi_brd-swapb.c`, `cnes_iasi_grd-swapb.ksh/cnes_iasi_grd-swapb.c`, `cnes_iasi_ctx-swapb.ksh/cnes_iasi_ctx-swapb.c`, `cnes_iasi_odb-swapb.ksh/cnes_iasi_odb-swapb.c`. A script `convert_config_files.ksh` may be used to check all the configuration files and convert them as necessary. Note that for OPS-LRS v6-0 onwards, the configuration files must be in big-endian format; for earlier versions they were required to be in native endian format.

Once the IASI PFS L1C has been generated, it is converted to an AAPP format to be ingested in the pre-processing step 2, `atovpp`. This task is done by `convert_iasi1c.ksh/convert_iasi1c.c`.

## 3.2. INTERFACES

Each step described above is followed by a reference level:

**Level 0:** HRPT data (NOAA) or PFS L0 (METOP): Raw telemetry data including house keeping and others raw data. Data of the different instruments are merged into a HRPT stream for NOAA. One file per instrument for METOP.

**AAPP level 1a:** separated data for each instrument

**AAPP level 1b:** Earth located and calibration coefficients (reversible: calibration coefficients are separated from raw data).

**AAPP level 1c:** Earth located and converted to brightness temperature data (non-reversible: calibration coefficients are applied to data)

**AAPP level 1d:** mapped and filtered data (with optional cloud mask in the case of HIRS).

**PFS level 1B (for AVHRR):** Earth located and calibration coefficients, flags.

**PFS level 1C (for IASI):** Gaussian-apodised, resampled radiance spectra, corrected for all geometrical and instrumental effects, with mapped AVHRR. Earth located.

**viiCT:** maia file (cloud products) for VIIRS (hdf5 format)

**avhCT:** maia file (cloud products) for AVHRR (hdf5 format)

For the NPP, JPSS and some other programmes (e.g. DMSP), NOAA adopt the following naming convention, and these names will be used in the AAPP documentation where applicable:

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**Raw data records (RDR):** Raw data from the instrument

**Temperature data records (TDR):** Calibrated, geolocated antenna temperatures from microwave sounder (i.e. no correction for antenna pattern). Original instrument grid.

**Sensor data records (SDR):** Calibrated, geolocated brightness temperatures, radiances or reflectivities. In the case of microwave instruments, antenna correction has been applied. Either original instrument grid or re-mapped.

**Environmental data records (EDR):** Geophysical quantities.

For NPP and JPSS programmes, AAPP ingests the SDRs. These are in one of two formats: (i) the HDF5 format defined by the JPSS Common Data Format Control Book [36], or (ii) a BUFR format whose contents closely reflects that of the HDF5 product.

### **3.3. DIAGRAMS**

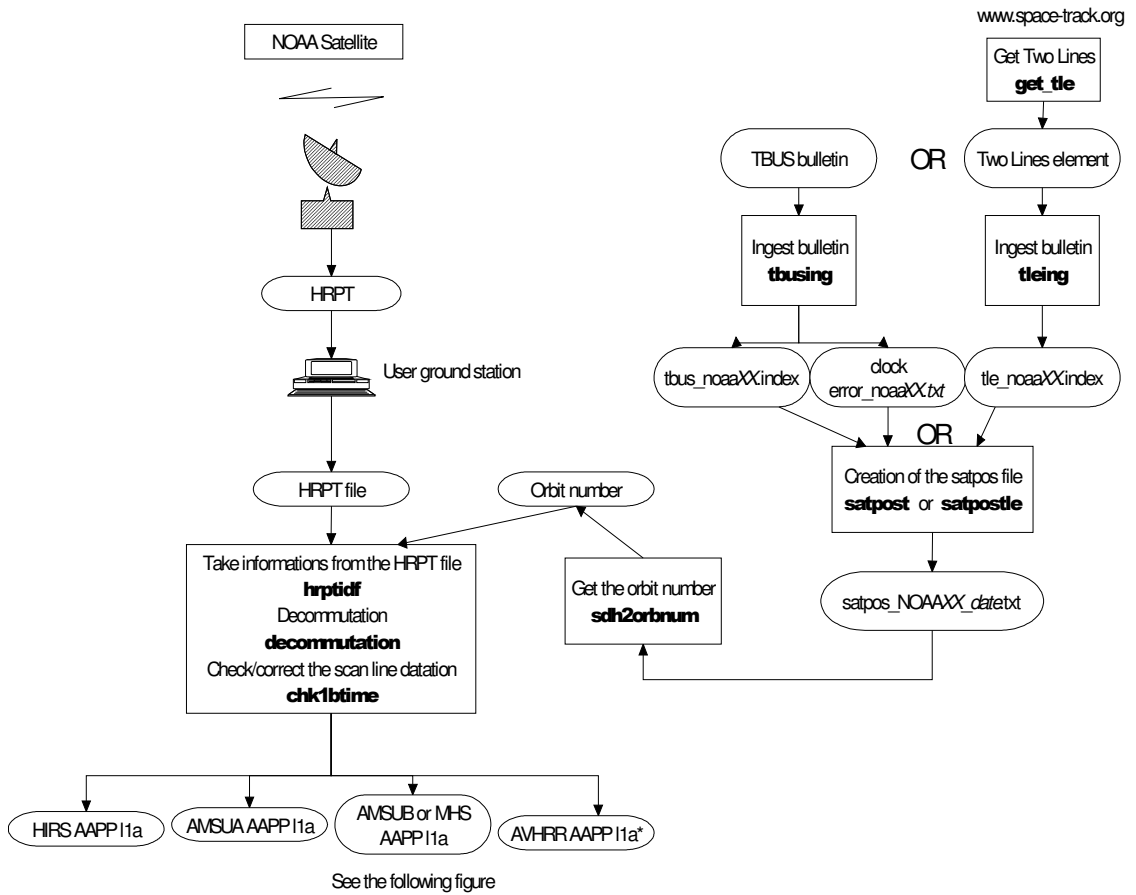
Different components of AAPP are used depending on the origin of the data.

In the following figures, the files that are created or modified by a process are noted. Summary files and fixed files are not noted.

### **3.4. DIRECT-READOUT OF NOAA SATELLITE DATA.**

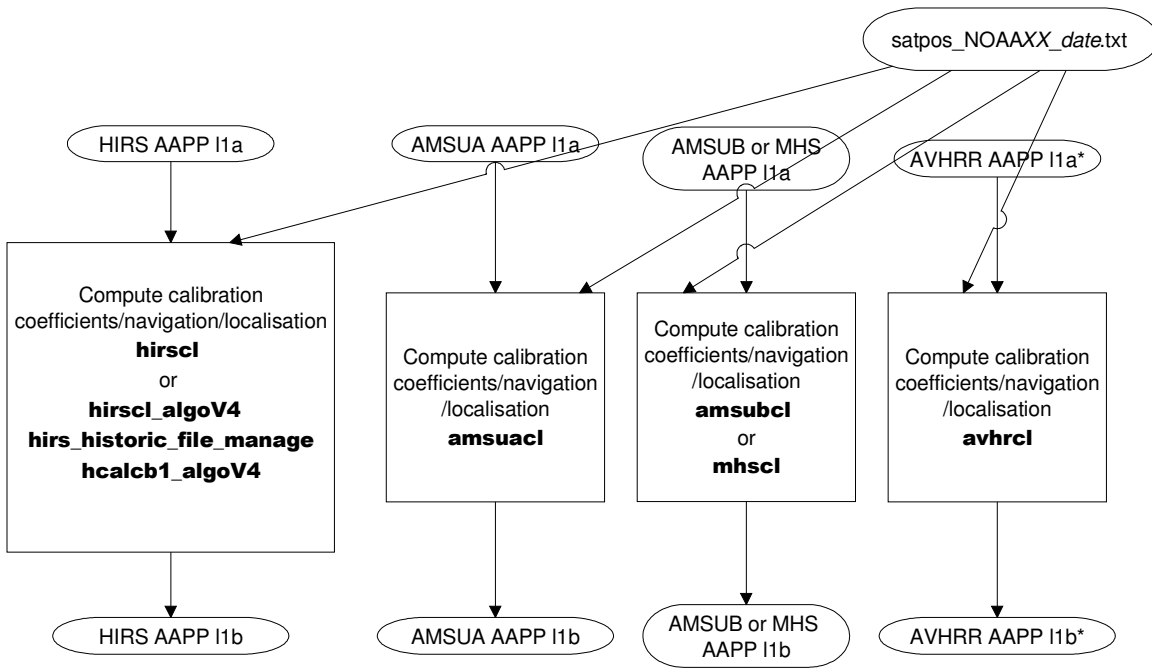
For NOAA direct readout, the interface to AAPP is at “Level 0”, i.e. the HRPT reception system is assumed to have the capability of receiving the NOAA HRPT data stream, as defined by NOAA [1].

**AAPP\_RUN\_NOAA** is the main module of the AAPP chain, for TOVS/ATOVS sounders and AVHRR radiometer on the NOAA satellites. It links up the different steps, ingest and pre-processing.



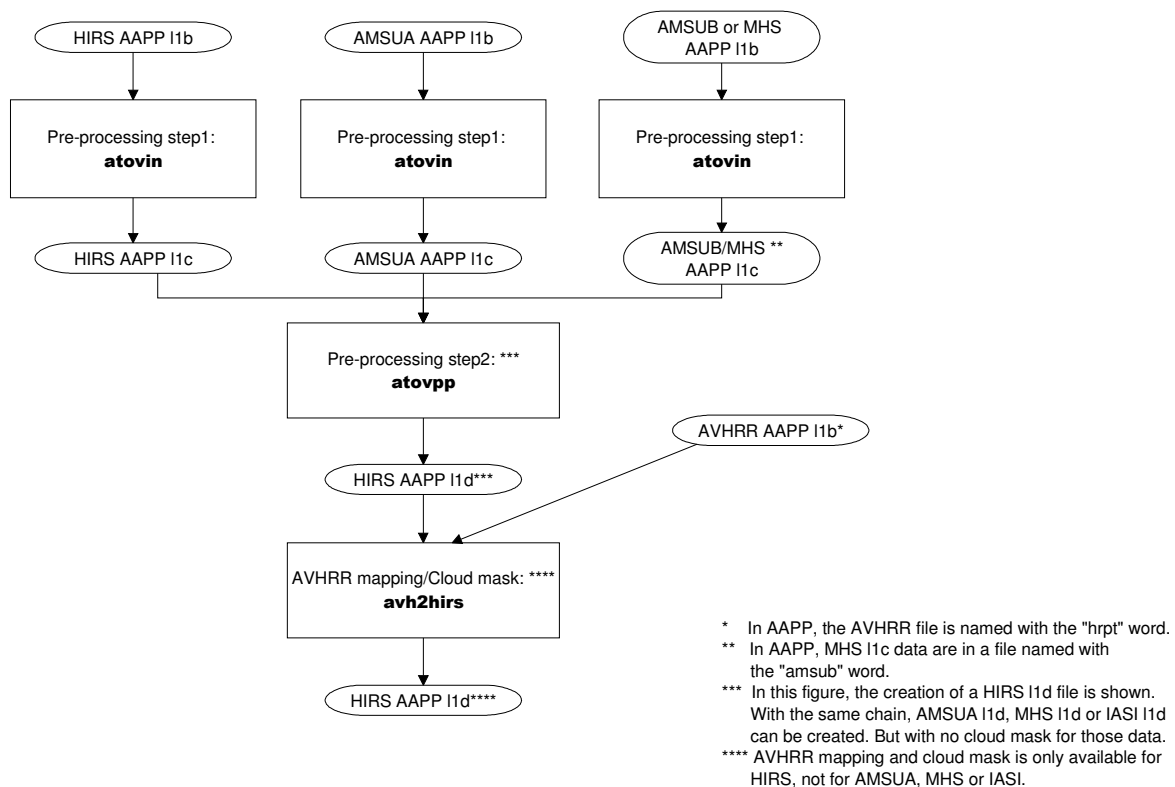
**Figure 3-1 : First steps for treating NOAA data**





See the following figure

**Figure 3-2 : Second steps for treating NOAA data**



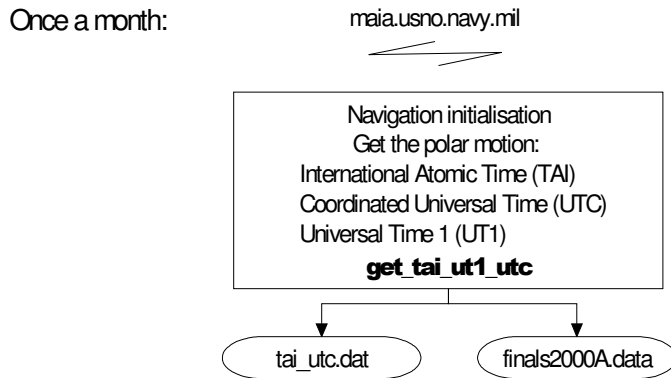
**Figure 3-3 : Pre-processing steps for NOAA data**

### **3.5. DIRECT-READOUT OF METOP SATELLITE DATA.**

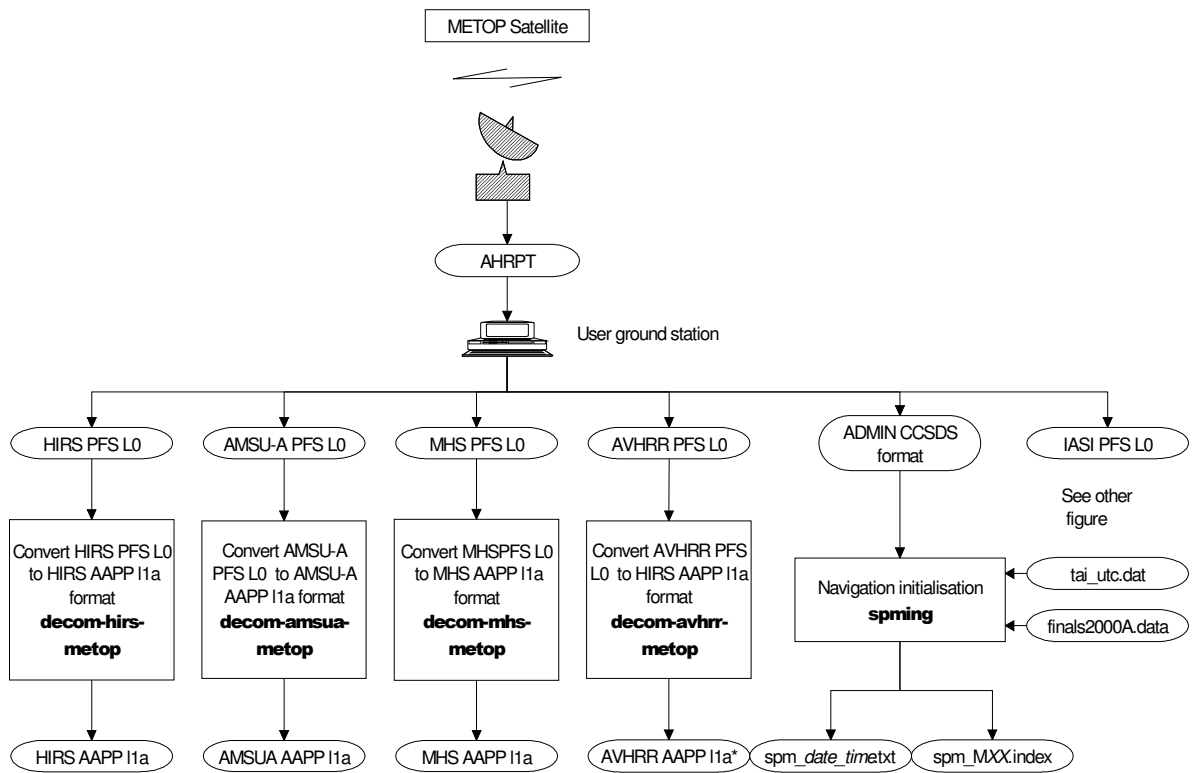
For METOP direct readout, the interface to AAPP is at “EPS Level 0”, i.e. the HRPT reception system is assumed to have the capability of receiving the METOP AHRPT data stream and converting to EPS level 0 format, as defined by EUMETSAT [25]. In this format the various instruments are delivered as separate files, therefore there is no need for a decommutation task.

Software tools are supplied within the “metop-tools” section of AAPP to convert EPS level 0 format to AAPP level 1a format. Calibration, navigation and pre-processing then proceed in the same way as for the NOAA satellites.

For a general description of the METOP processing, see the AAPP v6 (or v7) Top Level Design document [24]



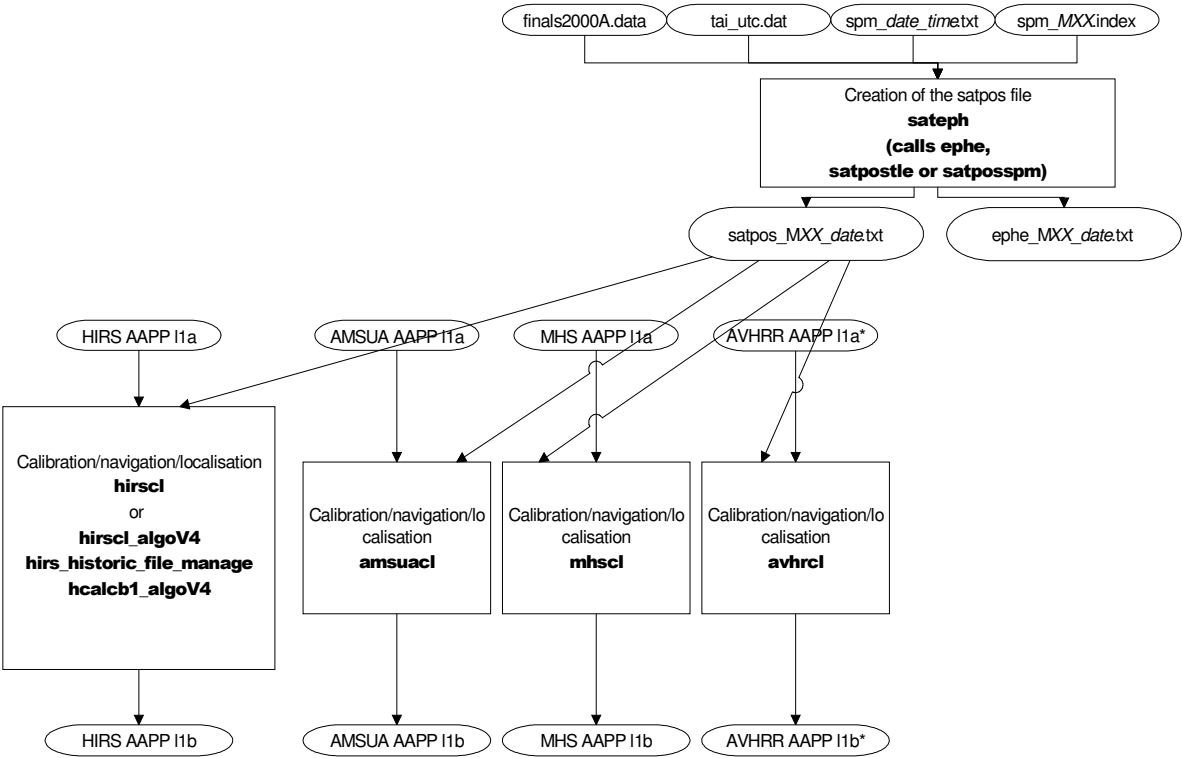
**Figure 3-4 : Periodical step for treating METOP data**



See the following figure

\* In AAPP, the AVHRR file is named with the "hrpt" word

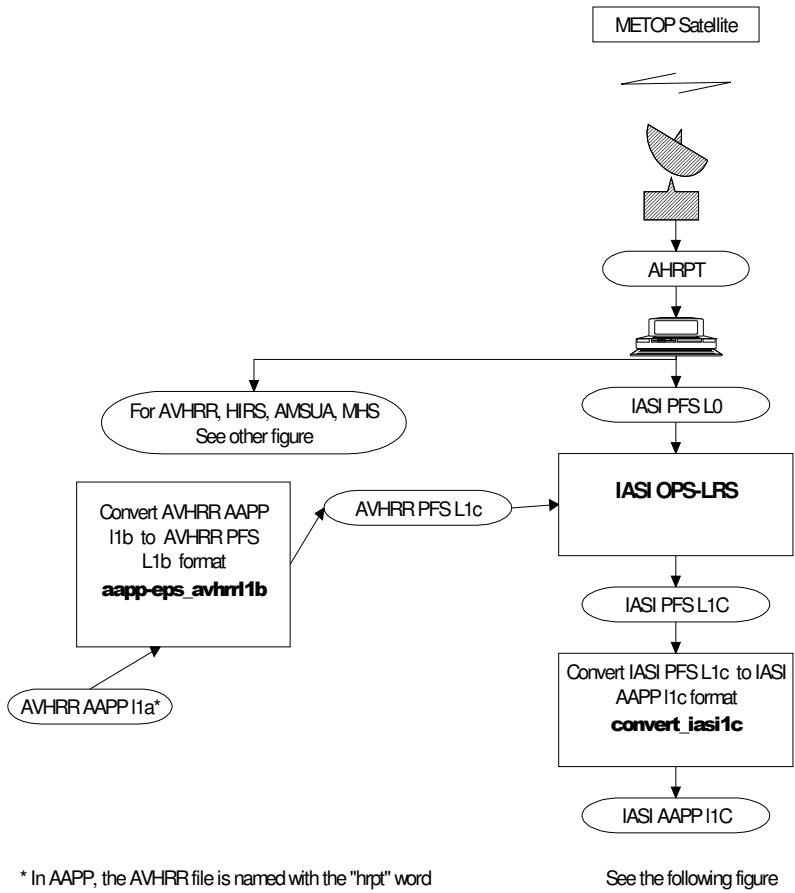
**Figure 3-5 : First steps for treating METOP data (ATOVS part)**



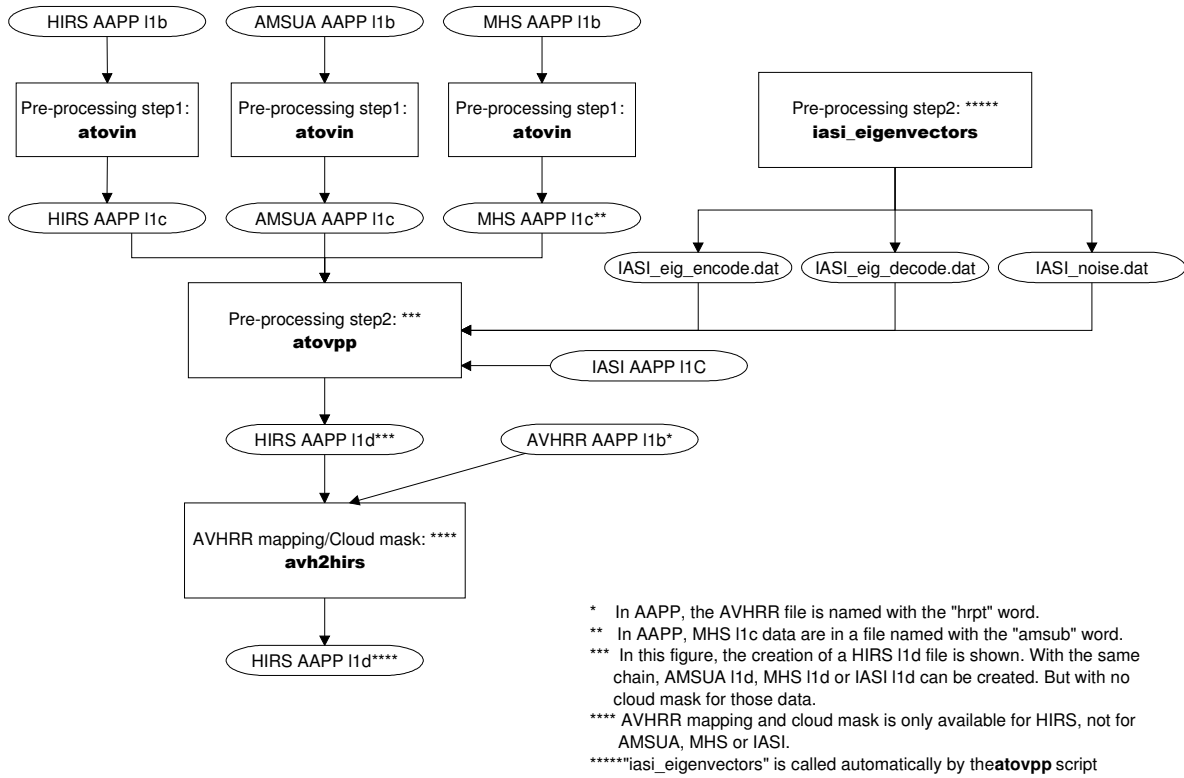
See the following figure

\* In AAPP, the AVHRR file is named with the "hrpt" word

**Figure 3-6 : Second steps for treating METOP data (ATOVS part)**

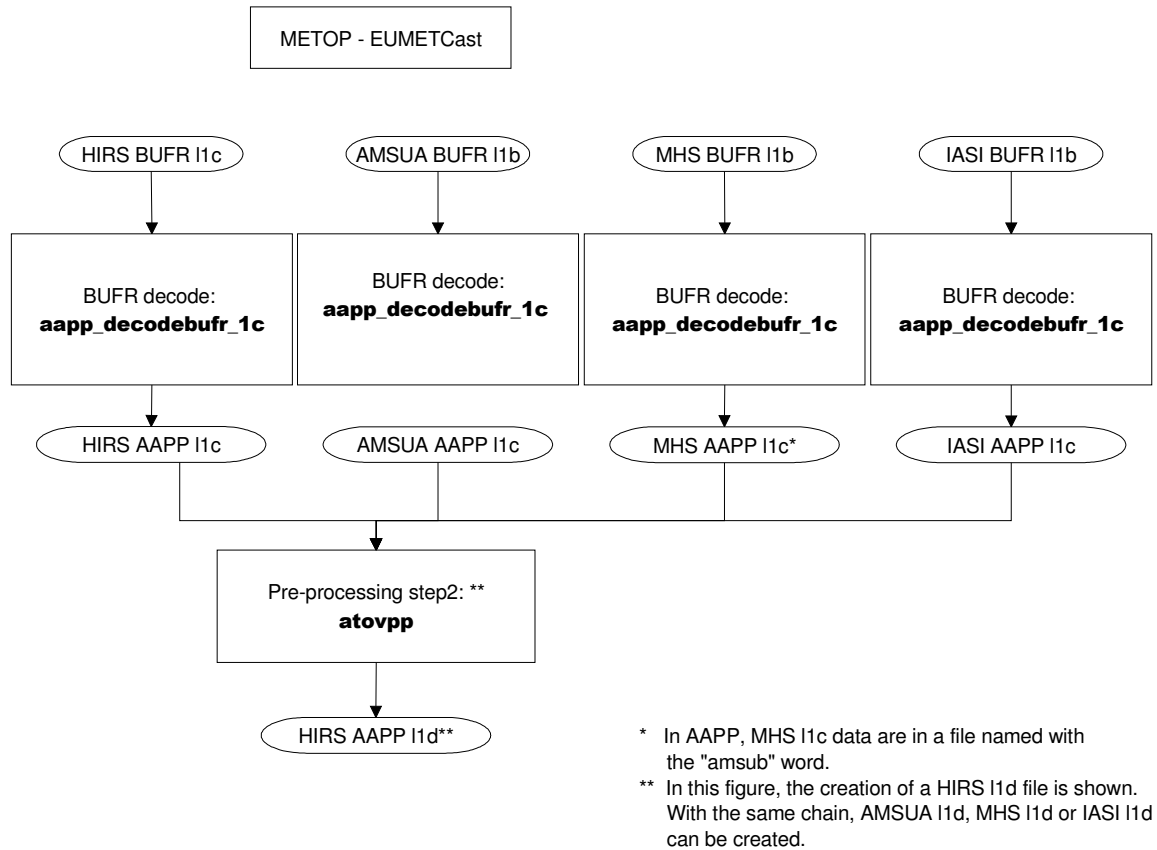


**Figure 3-7 : First steps for treating METOP data (IASI part)**

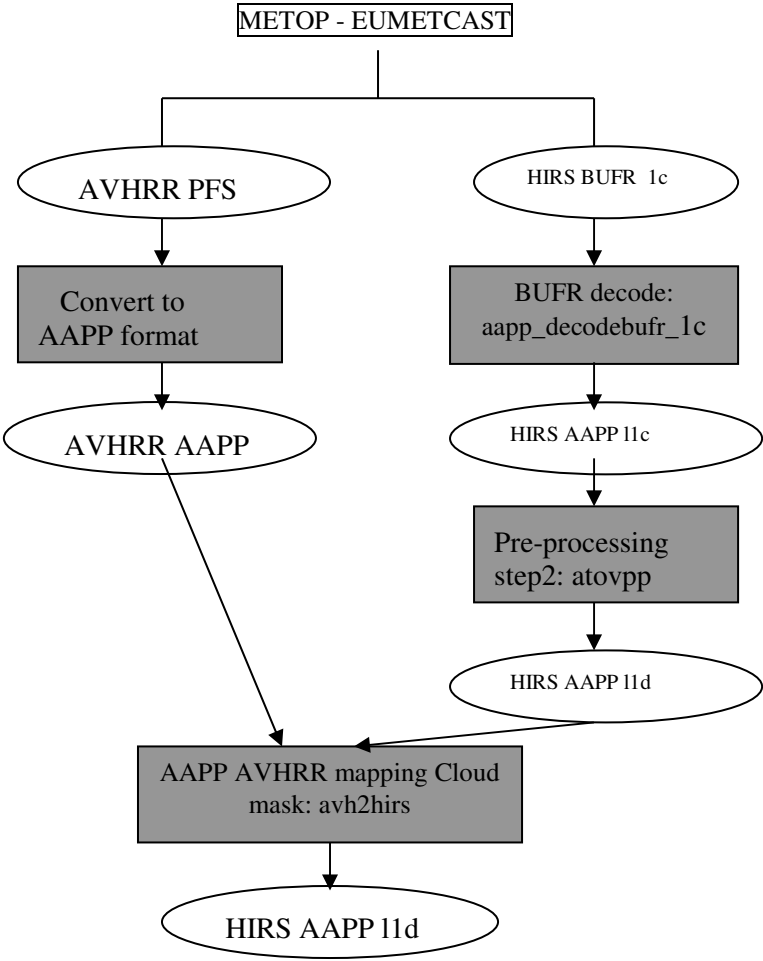


**Figure 3-8 : Pre-processing steps for METOP data**

### 3.6. ACQUISITION OF METOP DATA VIA EUMETCAST



**Figure 3-9 : Chain for treating METOP -ATOVS data received via EUMETCAST**



**Figure 3-10 : Chain for treating METOP –AVHRR - HIRS data received via EUMETCAST**



### 3.6.1. NOAA archived data

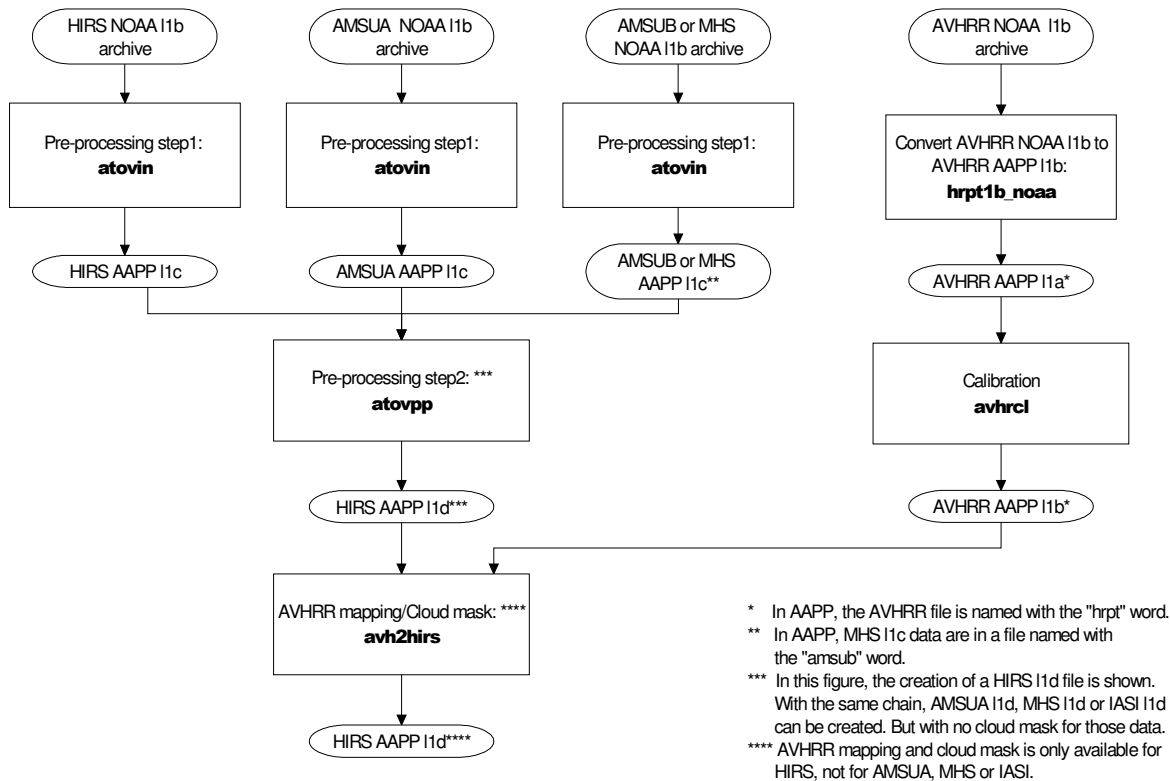


Figure 3-11 : Chain for treating archived NOAA data

## 4. GENERAL DESCRIPTION

### 4.1. SOFTWARE MAIN COMPONENTS

#### 4.1.1. Main module for direct-readout of NOAA satellites. AAPP\_RUN\_NOAA script

This module allows the user to link up the different steps of AAPP.

It receives as input the absolute pathname of the HRPT data file and the year of the data (this parameter is not present in the HRPT format).

With the tool **hrpidf.exe**, it extracts the satellite name, the day of the year and the time of the data.

The environment variables contained in the ATOVS\_ENV7 file determine the selection of the orbital bulletins and model. Two different bulletins and corresponding models can be selected: TBUS and Two-Line.

Case of TBUS:

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

By calling the module **tbusing** , it checks and ingests the TBUS bulletins useful to navigate the satellite.

It creates the satellite position-velocity file for several days (**satpos** file) with the command **satpost**.

Case of Two-Line:

By calling the module **tleing** , it checks and ingests the TLE bulletins useful to navigate the satellite.

It creates the satellite position-velocity file for several days (**satpos** file) with the command **satpostle**.

With the date, the time of the data and the satellite position file, it gets the orbit number (**sdh2orbnum**).

It distinguishes the pre-NOAA-K data (TOVS data) from NOAA-KLM data (ATOVS data) and from NOAA-N,N' data.

Then, it calls different modules to make the decommutation, navigation/localisation, calibration, mapping, cloud mask tasks (**decommutation, hirscl/hirscl\_algoV4, msucl, amsuacl, amsubcl, mhsccl, avhrel, atovin, atovpp, avh2hirs**).

For AVHRR, HIRS and MSU, before and after navigation/calibration task, AAPP\_RUN calls tools (**prhavh, prhirs, prhmsu**) to write level 1B headers and first records into ASCII files (*phavh\_before\_calib.log, phavh\_before\_calib.log, ...*).

At the end, it renames all output files to include information in the file names: Satellite name, date and time, orbit number.

#### 4.1.2. Main module for direct-readout of MetOp satellite. AAPP\_RUN\_METOP script

This module allows the user to link up the different steps of AAPP or AAPP/OPS-LRS.

All files to be processed are in a single directory

One file per instrument (i.e. dump mode)

File names follow the EUMETSAT convention, e.g.

```

AMSA_xxx_00_M04_20020808181206Z_20020808195406Z_N_O_20020808201206Z
MHSx_xxx_00_M04_20020808181201Z_20020808195401Z_N_O_20020808201201Z
HIRS_xxx_00_M04_20020808181200Z_20020808195358Z_N_O_20020808201200Z
AVHR_xxx_00_M04_20020808181200Z_20020808182359Z_N_O_20020808201200Z
HKTM_xxx_00_M04_20020808181200Z_20020808195358Z_N_O_20020808201200Z
IASI_xxx_00_M04_20020808181200Z_20020808195358Z_N_O_20020808201200Z

```

Two steps:

- a first one to get AMSU/HIRS/AVHRR products out
- a second one to run IASI OPS-LRS and generate products on IASI grid OPS-LRS requires AVHRR 11b. IASI OPS-LRS is not automatically included in the AAPP distribution. It must be requested by the user.

The environment variables contained in the ATOVS\_ENV8 file determine the selection of the orbital bulletins and model.

Case of TBUS:

By calling the module **tbusing** , it checks and ingests the TBUS bulletins useful to navigate the satellite.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

It creates the satellite position-velocity file for several days (**satpos** file) with the command **satpost**.

Case of Two-Line:

By calling the module **tleing** , it checks and ingests the TLE bulletins useful to navigate the satellite.

It creates the satellite position-velocity file for several days (**satpos** file) with the command **satpostle**.

With the date, the time of the data and the satellite position file, it gets the orbit number (**sdh2orbnum**).

Optionally, get OBT/UTC correlation parameters from Admin message in HKTM file and over-write VIADR in instrument files. (This step is not required if your station manufacturer has properly implemented the OBT-UTC handling).

Then, it calls different modules:

- . to convert in AAPP format 11b (**decom-amsua-metop**, **decom-mhs-metop**, **decom-hirs-metop**, **decom-avhrr-metop**)
- to make navigation/localisation, calibration (**hirscl/hirscl\_algoV4**, **msucl**, **amsuacl**, **amsubl**, **mhscl**, **avhrcl**).
- To make the preprocessing (**atovin**, **atovpp**, **avh2hirs**)

If OPS-LRS is present,

- OPS-LRS is called.
- the outfile is converted to AAPP 1C format
- the preprocessing **atovpp** and **avh2hirs** are called

At the end, it renames all output files to include information in the file names: Satellite name, date and time, orbit number.

#### 4.1.3. Main module for FY1 imager data. **AAPP\_RUN\_FY1** script

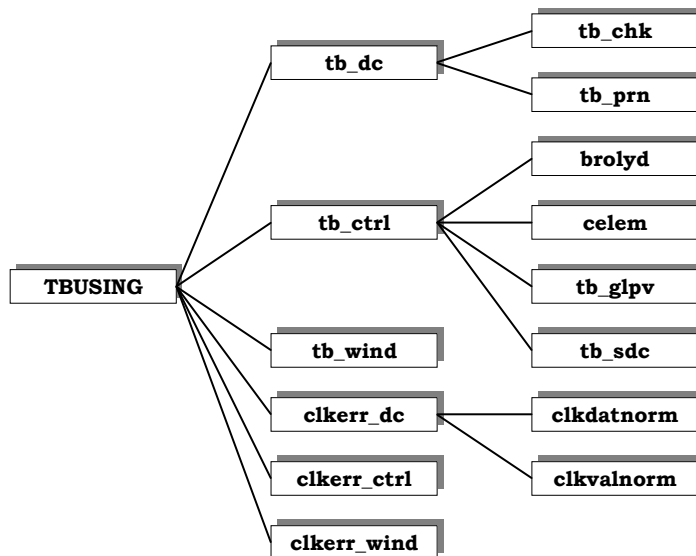
This section is deleted, as the FY-1D satellite is no longer operating.

#### 4.1.4. Satellite and image navigation initialisation: Ingest with **TBUS** bulletin, **TBUSING** script, **TBUSING.EXE** and satellite position and velocity: **SATPOST** script, **SATPOST.EXE**

*Modules TBUSING, TBUSING.EXE*

---

(See also reference manual pages: *tbusing.1* , *libtbus.3* , *tbus.5* , *clockerror.5* , *libbrolyd.3*)



**Figure 4-1 : Flow chart on the components of the TBUSING module**

These modules allow the ingest of TBUS bulletin(s). They can process one or several satellites (option). The TBUS file name can be specified (option). By default all the tbus files which are newer than the last update of the index files corresponding to the satellite list are ingested.

For each satellite, 2 historical files are created or updated:

- TBUS index file : relative to the TBUS orbital parameters. Each record contains epoch time, quality, tbus filename
- clock error file : contains all the clock error information which has been validated

The TBUS epoch may be at any position in the historical files which means that an old TBUS can be inserted in the files.

To insert new information :

- clock error and orbital parameter have to be extracted from TBUS resources bulletin.
- the user chooses files in relation to satellites to treat (input configuration).
- quality controls are made to check new orbit continuity compared to the preceding orbit (the **brolyd** extrapolation model is used), and to compare clock errors with the preceding ones.

#### TASK 1 : INPUT PARAMETERS READING

**tbusing** gets :

- Home directory of the TBUS files and bulletin(s) name(s) which will be stored in the TBUS index file.
- The list of satellites to be considered
- Historical file names

#### TASK 2: INITIALISATION

It opens the TBUS bulletin(s).

#### TASK 3: TBUS BULLETIN DECOMMUTATION AND VALIDATION TESTS

For each satellite:

It opens (or creates if files do not exist) the historical index file and the clock drift error file.

It calls different subroutines :

**tb\_dc** to decode the part IV of the TBUS bulletin to extract orbital parameters and to check that extracted parameters are in the authorised value area.

**tb\_ctrl** to check the orbital parameters continuity (to compare them with the last valid parameters registered in the historical file), using the **brolyd** extrapolation model. The new TBUS file is declared OK if the errors are less than 6 km/day. The tests with the last preceding valid tbus are done only if the time difference is less than 7 days.

**tb\_wind** to write the valid TBUS information record at the end of the historical file.

**clkerr\_dc** to decode the clock error values stored in the plain language message at the end of the TBUS file Part IV and to check that extracted clock errors are in the authorised values area.

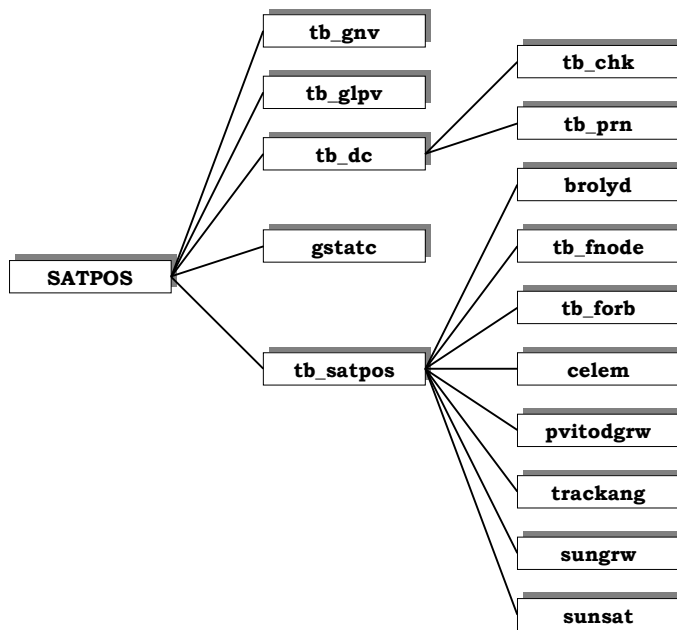
**clkerr\_ctrl** to check the decoded clock values by comparing them to the preceding historical values.

**clkerr\_wind** to write the valid clock error information record at the end of the clock drift data file, and on the standard input.

At the end, tusing closes the different files.

***Modules SATPOST ,SATPOST.EXE***

(See also reference manual pages: *satpost.1* , *satpos.5* , *libbrolyd.3*)



**Figure 4-2 : Flow chart on the SATPOST module components.**

These modules create a satellite position-velocity file (satpos file) for a given satellite, for a given station, a start time and a given duration. They search the TBUS bulletin file for the orbital parameters time closest to the given start time.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### TASK 1: INPUT PARAMETERS READING

**satpost** gets :

- The satellite name and the station name
- The start time from which the orbital parameters are extrapolated.
- The time step and the number of days.
- The home directory for the TBUS files and the index file name.
- The criteria to search the TBUS bulletin (the nearest or the preceding one).

#### TASK 2: INITIALISATION

It finds, opens and reads the TBUS bulletin corresponding to the research criterion.

To find the file name of the valid TBUS bulletin, it calls the subroutine **tb\_gnv** if the search criteria is the nearest to the start time. The searched TBUS date must be in a time interval. It calls **tb\_glpv** if the search criterion is the last preceding valid TBUS filename from the index file. The index file is supposed to be chronological

**tb\_dc** decodes the part IV of the TBUS bulletin to extract orbital parameters and to check that extracted parameters are in the authorised value area.

By calling **gstatc**, it initialises the station coordinates (latitude, longitude, altitude) from the file *stations.txt* (directory DIR\_STATIONS defined in ~/ATOVS\_ENV) and then converts them into Greenwich cartesian coordinates.

**satpost** returns information on standard output.

#### TASK 3: POSITION CALCULATIONS FOR ALL THE STEPS

**tb\_satpos** does this task. It calculates the satellite position. The calculations are made since the start date during several days with a time increment. It begins by initialising the **brolyd** model with the current TBUS.

For each time the following calculations are performed (calculation loop):

the satellite position and velocity in the inertial reference frame using the **brolyd** extrapolation model.

conversion into a Greenwich reference frame (**celem** and **pvitodgrw**).

orbit number deduced from the z component

visibility from the station including refraction (**trackang**)

satellite in daylight or nighttime conditions if the satellite is seen from the station (**sungrw**, **sunsat**).

It writes the results on the standard output.

**tb\_satpos** calls others subroutines to initiate variables useful to **brolyd** model:

- **tb\_fnode** calculates nodal period (time interval between 2 successive ascending nodes) and ascending node time of the first orbit after the TBUS date.
- **tb\_forb** calculates the orbit number for the given date (from the nodal period and the initial ascending node time).

#### **4.1.5. Satellite and image navigation initialization with Two Line Element sets: GET\_TLE script, TLEING script, TLEING.EXE and satellite position and velocity: SATPOSTLE script, SATPOSTLE.EXE.**

##### Module GET\_TLE

(See also reference manual pages: *get\_tle.1* *tleing.1* , *tle.5* )

This script allows the retrieval of the most recent Two-Line bulletin(s) (tle) from the web site Space-Track or Celestrak, or another source, using the GNU command **wget**.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

The default connection (set in the ATOVS\_ENV8 file) is to retrieve TLEs from EUMETSAT. All parameters are configured in the ATOVS\_ENV8 file and are self documented. For retrieval from space-track, a username and password are required that must be requested individually by the user to the Space-Track web site.

TASK 1 : INPUT PARAMETERS READING

**get\_tle** gets :

- Home directory of the TLE files.
- The URL for login
- The URL for download
- The user name and password for Space-Track connection.
- The time-out for connections.

TASK 2: LOGIN

Sends a **wget** commands that logins and store cookies in a temporary file (for space-track only)

TASK 3: DOWNLOAD AND STORE

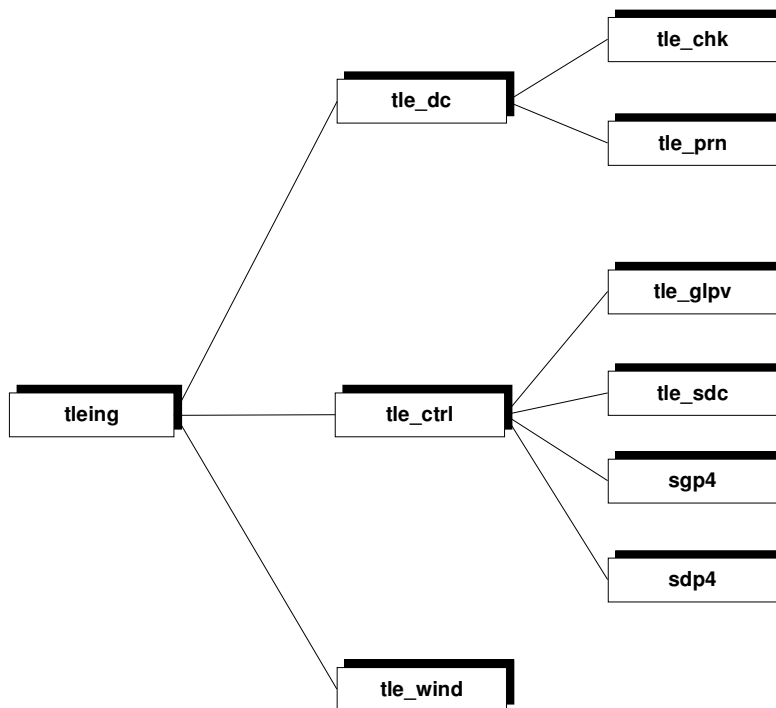
Sends a **wget** command to download selected file, and load login cookies (if applicable)

Uncompress the file with **gunzip** command, if applicable

Store file in TLE directory

Modules TLEING, TLEING.EXE

(See also reference manual pages: *tleing.1* , *libtle.3* , *tle.5* , *libsgp.3f*)



**Figure 4-3 : Flow chart on the components of the TLEING module**

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

These modules allow the ingest of Two-Line bulletin(s) (tle). They can process one or several satellites (option). The Two-Line file name can be specified (option). By default all the tle files which are newer than the last update of the index files corresponding to the satellite list are ingested.

For each satellite, one historical file is created or updated:

- TLE index file: relative to the TLE orbital parameters. Each record contains epoch time, quality, tbus filename

The TLE epoch may be at any position in the historical files which means that an old TLE can be inserted in the files.

To insert new information:

- orbital parameters have to be extracted from TLE resources bulletin.
- the user chooses files depending on which to satellites are to be processed (input configuration).
- quality controls are made to check new orbit continuity compared to the preceding orbit (the **sgp4** extrapolation model is used).

#### TASK 1 : INPUT PARAMETERS READING

**tleing** gets:

- Home directory of the TLE files and bulletin(s) name(s) which will be stored in the TLE index file.
- The list of satellites to be considered
- Historical file names

#### TASK 2: INITIALISATION

It opens the TLE bulletin(s).

#### TASK 3: TLE BULLETIN DECOMMUTATION AND VALIDATION TESTS

For each satellite:

It opens (or creates if files do not exist) the historical index file.

It calls different subroutines :

**tle\_dc** to decode the TLE bulletin to extract orbital parameters and to check that extracted parameters are in the authorised value area.

**tle\_ctrl** to check the orbital parameters continuity (to compare them with the last valid parameters registered in the historical file), using the **sgp** extrapolation model. The new TLE file is declared OK if the errors are less than 6 km/day. The tests with the last preceding valid tbus are done only if the time difference is less than 7 days.

**tle\_wind** to write the valid TLE information record at the end of the historical file.

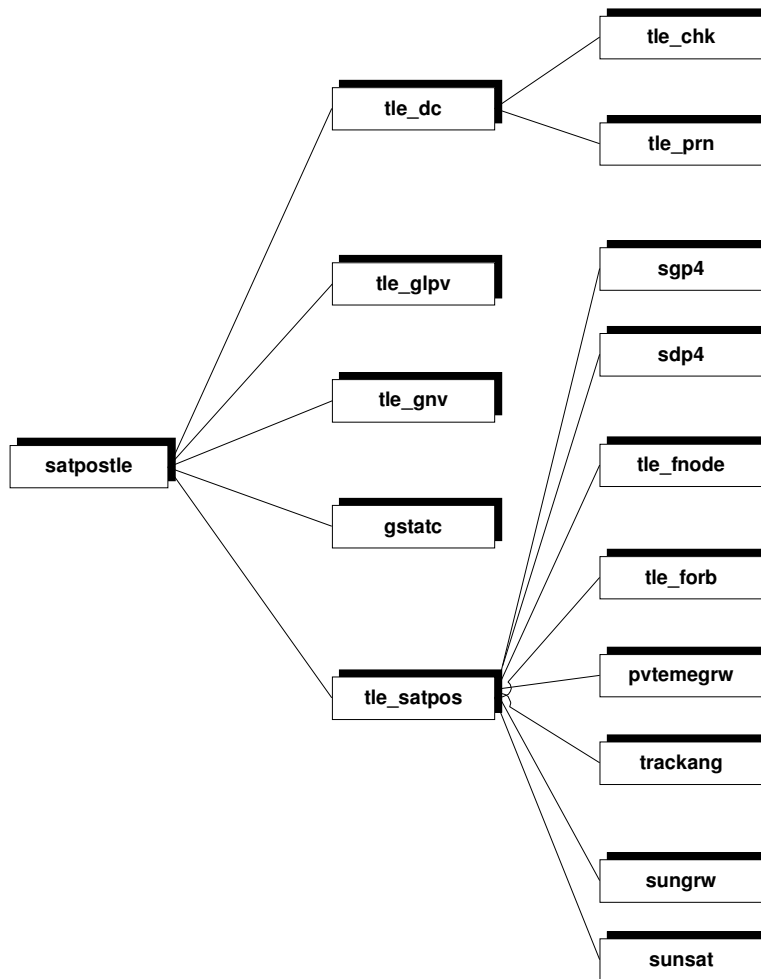
At the end, tleing closes the different files

---

#### ***Modules SATPOSTLE, SATPOSTLE.EXE***

(See also reference manual pages: *satpostle.1* , *satpos.5* , *libsgp.3f*)





**Figure 4-4 : Flow chart on the SATPOSTLE module components.**

These modules create a satellite position-velocity file (satpos file) for a given satellite, for a given station, a start time and a given duration. They search the TLE bulletin file for the orbital parameters time closest to the given start time.

**TASK 1: INPUT PARAMETERS READING**

**satpostle** gets :

- The satellite name and the station name
- The start time from which the orbital parameters are extrapolated.
- The time step and the number of days.
- The home directory for the TLE files and the index file name.
- The criteria to search the TLE bulletin (the nearest or the preceding one).

**TASK 2: INITIALISATION**

It finds, opens and reads the TLE bulletin corresponding to the research criterion.

To find the file name of the valid TLE bulletin, it calls the subroutine **tle\_gnv** if the search criteria is the nearest to the start time. The searched TLE date must be in a time interval. It calls **tle\_glpv** if the search criterion is the last preceding valid TLE filename from the index file. The index file is supposed to be chronological

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**tle\_dc** decodes the TLE bulletin to extract orbital parameters and to check that extracted parameters are in the authorised value area.

By calling **gstatc**, it initialises the station coordinates (latitude, longitude, altitude) from the file *stations.txt* (directory DIR\_STATIONS defined in ~/ATOVS\_ENV) and then converts them into Greenwich Cartesian coordinates.

**satpostle** returns information on standard output.

#### TASK 3: POSITION CALCULATIONS FOR ALL THE STEPS

**tle\_satpos** does this task. It calculates the satellite position. The calculations are made since the start date during several days with a time increment. It begins by initialising the **sgp4/sdp4** model with the current TLE.

For each time the following calculations are performed (calculation loop):

the satellite position and velocity in the inertial reference frame using the **sgp4/sdp4** extrapolation model.

conversion into a Greenwich reference frame (**pvtmegrw**).

orbit number deduced from the z component

visibility from the station including refraction (**trackang**)

satellite in daylight or nighttime conditions if the satellite is seen from the station (**sungrw, sunsat**).

It writes the results on the standard output.

**tle\_satpos** calls others subroutines to initiate variables useful to **sgp** model:

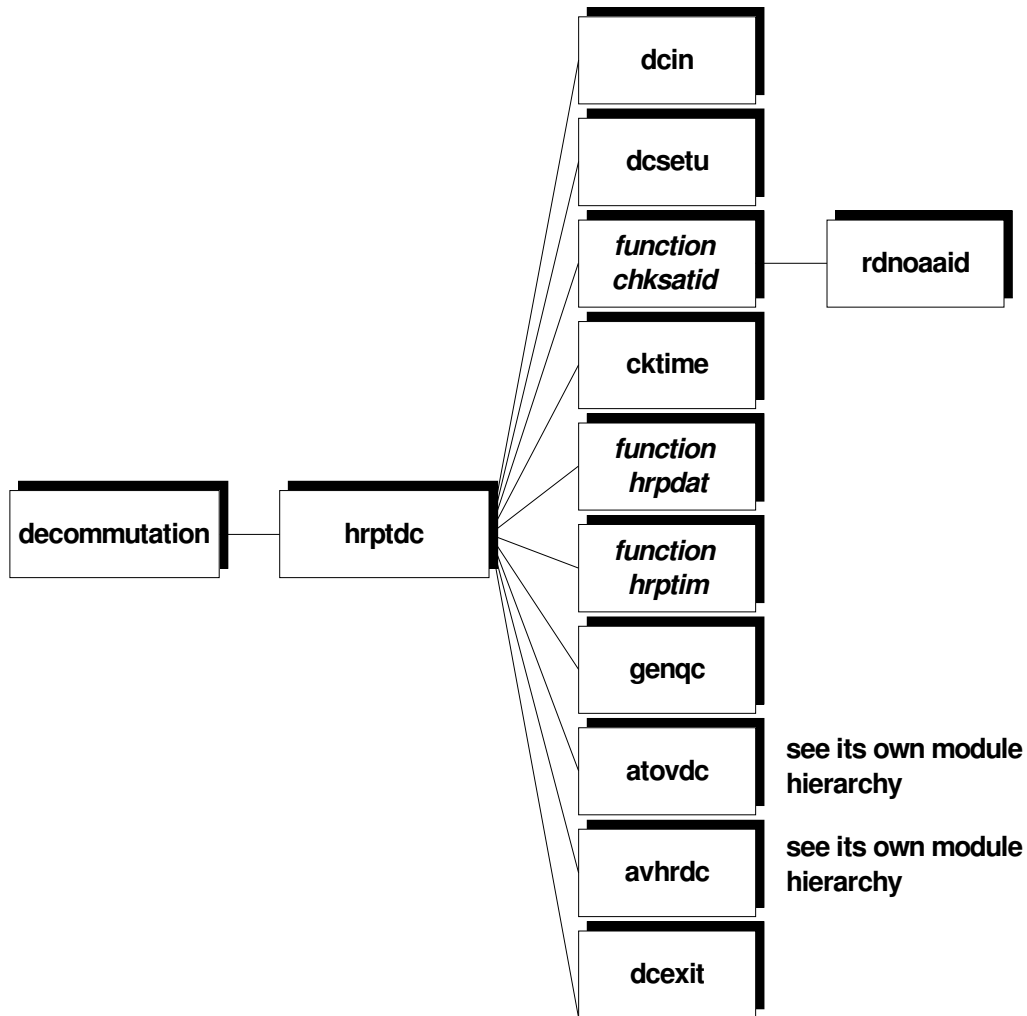
- **tle\_fnode** calculates nodal period (time interval between 2 successive ascending nodes) and ascending node time of the first orbit after the TLE date.
- **tle\_forb** calculates the orbit number for the given date (from the nodal period and the initial ascending node time).

---

#### ***Modules MMAM.EXE and PATCH-LEVEL0-FROM-MAMM.EXE***

These modules are used for Metop admin messages (HKTM files). The decoding is done by *mamm-main.exe*; this program extracts the MAMM bulletin from binary ADMIN messages and outputs an ASCII representation (XML format). Time parameters can then be extracted from the XML file and used for patching the VIADR record in METOP PFS level 0 files with *patch-level0-from-mamm.exe*

#### **4.1.6. Decommuration modules: DECOMMUTATION script and DECOMMUTATION.EXE.**

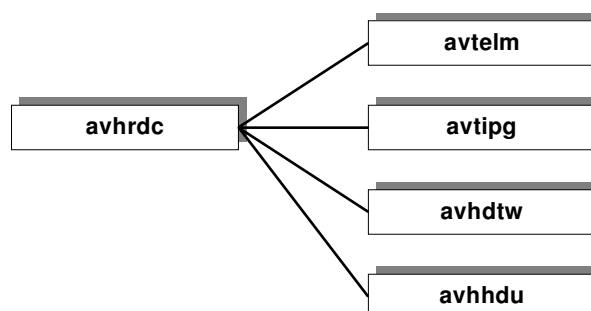


**Figure 4-5 : DECOMMUTATION and HRPTDC module hierarchy.**

To simplify the diagram, the calls to subroutines of the **libf7ml** , **libf7tp** , **libf7nl1b** libraries have not been written.



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---



**Figure 4-7 : AVHRDC components hierarchy.**

To simplify the diagram, calls to the subroutines of the libraries like **libf7ml**, **libf7gp** have not be written

Decommutation modules perform the extraction task for several parts of the HRPT stream (level 0) which have to be processed by **avhrdc** (AVHRR decommutation task) and **atovdc** (TOVS/ATOVS decommutation task). The HRPT minor frames, numbered 1 to 3, are received by the center- specific routines and can be processed in real time from several local acquisition systems or read off-line from files coming from various centers. The HRPT minor frames are read by a center- specific routine. This is necessary as the extract format of the HRPT minor frames will depend on the hardware of the reception station.

After decommutation, there is one raw data file for each instrument. Those files represent the level 1a data.

#### TASK 1: INITIALISATION

**hrptdc** performs this task.

This module receives as an input the unpacked HRPT minor frame(s) coming from the center specific module closely connected to the hardware. The HRPT minor frame is an array of 11090 words made of the 10 bits HRPT words placed right justified in 16 bits words. **hrptdc** detects the end of HRPT stream.

**hrptdc** reads input options (**dcin**) and opens the various files (**dcsetu**).

It identifies the satellite (**chksatid**) by checking HRPT and TIP satellite Id coherence.

It checks good start condition: an HRPT minor frame equal to 1, with valid time and good time difference between consecutive HRPT minor frames (**cktime**). This means that under normal circumstances a few minor frames at the start of the pass will not be processed, as they are used for consistency checking. If it is known that there is no bad data at the start (e.g. when processing granules) then the consistency check can be disabled by setting an environment variable (**SKIP\_DECOM\_CHECK=Y**); in this case processing will start at the first minor frame number 1.

It performs general quality controls for one HRPT minor frame (**genqc**).

- Check the satellite identification at the first call
- Check of the number of the scan line
- Check of the date and time
- Check of the minor frame number
- Check of the TIP parity bits in the five consecutive TIP minor frames
- Check parity bits in every TIP word for ATOVDC and flag the relevant bits in the quality indicator

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

It computes the number of possible missing HRPT minor frames (= the number of possible missing AVHRR scan line).

It calls the routine **atovdc** that will extract HIRS, AMSU-A/B (or MSU if TOVS, MHS if NOAA-N,N') and DCS data from TIP/AMSU minor frames. TIP/AMSU minor frames are embedded in 3 consecutive HRPT minor frames. The first one contains 5 TIP minor frames, the second one contains 'backfill' (dummy data) and the third one contains 5 AMSU minor frames. For pre-NOAA-K satellite, each of the 3 consecutive HRPT minor frames contains the same 5 TIP minor frames.

It calls the routine **avhrdc** that extracts AVHRR data from one HRPT minor frame.

#### TASK 2: TOVS/ATOVS AND DCS DECOMMUTATION TASK

The module **atovdc** performs this task called by the **hrptdc**.

It receives as input from **hrptdc**:

- 5 TIP or AMSU minor frames (extracted from one HRPT minor frame).
- HRPT minor frame number (1,2 or 3)
- The number of the HRPT minor frame in the orbit (= AVHRR scan line in the orbit).
- The number of missing HRPT minor frame
- The satellite identification
- Various dates and times
- ...

It determines if minor frames contain pre-NOAA-K data or not from the satellite ID, at the first call.

It removes 2 least significant parity bits: TIP/AMSU words are 8 bits words, HRPT words are 10 bits words.

It determines if minor frames contain TIP, AMSU or backfill data.

If TIP minor frame (=if HRPT minor frame number equal to 1) :

- Extracts TIP minor frame counter and TIP major frame counter (**getmf**)
- Extracts time from TIP word in TIP minor frame number 0 (**tiptim**)
- Performs quality controls (**tipqc**)
- Calls the routine **hirget** that extracts HIRS/3 words
- If pre-NOAA-K data, calls the routine **otiget** that extracts HIRS/2 and MSU words
- Calls the routine **dcsgt** that extracts DCS words
- Calls the routine **hirout** when the HIRS scan line is full
- If pre-NOAA-K data, calls the routine **msuout** when the MSU scan line is full
- Calls the routine **dcnout** when DCS data is full

If AMSU minor frame (=if HRPT minor frame number equal to 3) :

- Extracts minor frame counter to find good conditions to start
- Performs quality controls (**amsuqc**)
- Calls the routine **amsget** that extracts AMSU-A1/A2 and B (or MHS) words
- Calls the routine **amsout** when the AMSU scan line is full

If HRPT minor frames are missing, it fills arrays.

If TIP minor frame, **atovdc** extracts analog housekeeping telemetry data (**anaget**).

Finally, on completion of the **atovdc** module, separated HIRS/3 (or HIRS/4), AMSU-A, AMSU-B (or MHS) & DCS level 1a files are obtained, or HIRS/2, MSU and DCS level 1a files in case of pre-NOAA-K TIP data.

#### TASK 3: AVHRR DECOMMUTATION TASK

The module **avhrdc** performs this task, called by **hrptdc**.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

It receives as input from **hrptdc**:

- an array of HRPT minor frames (in actual fact this array contains only one HRPT minor frame in this version of AAPP).
- Miscellaneous variables : The minor frame number of the orbit (=the number of the AVHRR scan line), the number of missing HRPT minor frames (=the number of missing AVHRR scan lines), the HRPT minor frame number(=1 or 2 or 3), the satellite identification, the orbit number, dates and times,...

It fills the variables for one record of the AVHRR output file (=for one AVHRR scan line) :

- Variables of the scan line information part.
- Date and time
- Quality indicators from **genqc** results.
- Variables of the telemetry data part (**avtelm**).
- Variables of the video data.
- Variables of the TIP header data part and the CPU A and B telemetry part from TIP data (**avtipg**).

It calls the routine **avhdtw** which writes the direct access AVHRR output file, corresponding to the given scan line number.

It updates the header variables in the avh1bdh common (**avhhdh**).

#### TASK 4: CORRECT SCAN LINE DATATION FOR LEVEL 1 B FILES

(See also reference manual pages *chk1btime.1*)

The module **chk1btime** checks and corrects the scan line datation for a given level 1a file that has been processed by **atovdc**. **chk1btime** is called for HIRS, MSU, AMSU-A and AMSU-B instruments. The AVHRR (hrpt.11b file) does not require **chk1btime** correction.

*Note that **chk1btime** can not work for NOAA level 1b file because NOAA files have missing records. AAPP ones do not have missing records because AAPP fills records when scan lines are missing.*

The error in **atovdc** is to use the same date information for all instruments. The AAPP developers have preferred to correct the files than fixing the bug in the decommutation step.

This program is dependant on 1B format structure.

It trusts the time indicated in the 1st scan line of the file.

#### **4.1.7. EPS level 0 to AAPP level 1a conversion for METOP: DECOM-AMSUA-METOP script and AMSUA-MAIN.EXE, DECOM-MHS-METOP script and MHS-MAIN.EXE, DECOM-AMSUA-HIRS script and HIRS-MAIN.EXE, DECOM-AVHRR-METOP script and AVHRR-MAIN.EXE.**

These modules are to be found in the “metop-tools” directories. Each script takes two arguments: the name of the Level 0 input file and the name of the level 1a output file.

There is a script and a binary program associated with each instrument.

<i>Instrument</i>	<i>Script</i>	<i>Binary</i>
HIRS	decom-hirs-metop	hirs-main.exe
AVHRR	decom-avhrr-metop	avhrr-main.exe

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002
		Version : 8.1 Date : March 2019

<i>Instrument</i>	<i>Script</i>	<i>Binary</i>
MHS	decom-mhs-metop	mhs-main.exe
AMSUA	decom-amsua-metop	amsua-main.exe

In order to process level 0 data, each main program implements five routines; for instance, hirs-main.c contains the definition of the following routines:

- hirs\_11b\_open
- hirs\_11b\_write\_header
- hirs\_11b\_write\_record
- hirs\_11b\_write\_dummy
- hirs\_11b\_close

Each of these routines call the Fortran routines of AAPP; we list here what those routines are for each instrument:

	<i>AVHRR</i>	<i>HIRS</i>	<i>MHS</i>	<i>AMSUA</i>
11b_open	Fortran open	Fortran open	Fortran open	Fortran open
11b_write_header	avhhdw	hrshdw	mhshdw	amahdw
11b_write_record	avhrdc	hirout	amsout	amsout
11b_write_dummy	avhrdc	hirout	amsout	amsout
11b_close	Fortran close	Fortran close	Fortran close	Fortran close

In common-main.c, the main loop for level 0 processing is implemented (subroutine common\_loop). This processing loop is used for AVHRR, HIRS and MHS. AMSUA data processing requires its own loop, because of the two sub-instruments AMSUA1 and AMSUA2.

The processing loop reads level 0 data using the library libeps\_metop10 and passes instrument data packets to AAPP using the five routines described above.

CCSDS packets are decoded using libccsds, and UTC time is computed from OBT using libobtutc.

AAPP libf7tp and libsatic are used too.

#### **4.1.8. Convert chrpt (FY1c and FY1d satellites) to hrpt (NOAA satellites): convert\_chrpt script and convert\_chrpt.exe**

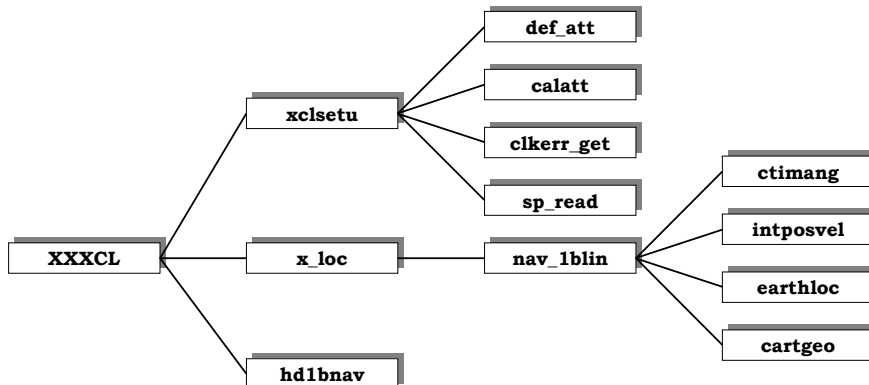
This section is deleted as the FY1 satellites are no longer operating.



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**4.1.9. Image navigation modules: HIRSCL script and HIRSCL.EXE, HIRSCL\_ALGOV4 script and HIRSCL\_ALGOV4.EXE, MSUCL script and MSUCL.exe, AMSUACL script and AMSUACL.EXE, AMSUBCL script and AMSUBCL.EXE, MHSCL script and MHSCL.EXE, AVHRCL script and AVHRCL.EXE.**

(See also reference manual pages: *libnavnoaa.3*, *libnavtool.3*, *libsatid.3* and detailed navigation equations in [17] )



**Figure 4-8 : general flow chart on the location module components :  
HIRSCL/MSUCL/AMSunCL/MHSCL/AVHRCL**

The image navigation converts the line and pixel into latitude and longitude for any pixel of the image. The task needs files: the level 1a file of the considered instrument, the SATPOS file, the CLOCK ERROR file. Modules are called for the different tasks.

TASK 1 : INITIALISATION

**hclin/hclin\_algoV4/mclin/amaclin/ambclin/mhsclin/avhclin** get logical units of the files.

**hclsetu/mclsetu/amasetu/amsubsetu/mhssetu/avhsetu :**

- Open and read the level 1a file.
- Open the SATPOS file.
- **sp\_read** reads the SATPOS file and tests if the input starting time is included into the SATPOS file, and if the satellite Id and memory are correct.
- If input attitude is missing, call **def\_att** that returns the default attitudes value. Those values depend on satellite (see satid file).
- Open the CLOCK ERROR file.
- Call **calatt** that calculates the attitude error matrix for small yaw, roll and pitch angles. This matrix is allowed to change the reference: local orbital coordinates (Rv) (x: satellite vertical, y: normal to x and z, z: normal to x and to velocity vector) / coordinates (Rs) linked to the spacecraft structure.
- Get clock error data if the level 1a has not already been taken into account in level 1a (**clkerr\_get**)
- Initialise navigation parameters.

TASK 2: CALCULATION OF THE IMAGE NAVIGATION PARAMETERS

**h\_loc/m\_loc/ama\_loc/amb\_loc/mhs\_loc** check if the clock error has already been applied. If not applied, the time and clock flag control of every line of data are modified and updated for level 1b. They call the routine **nav\_11blin**.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**nav\_1blin** computes the navigation variables of the level 1b, for one scanning line and for one instrument number. It calls the following routines and functions: **genattid**, **lptoviewvect**, **intposvel**, **snagre**, **earthpix**. All information on default attitude, misalignment and description of instruments scanning functions is stored in a satellite identification file (see satid.5 libsatid.3)

**genscid** and **genattid** returns the nominal attitude mode of the satellite. The different attitude modes that can be considered are:

- local normal pointing mode
- yaw steering mode
- geocentric mode

**lptoviewvect** converts the line and pixel numbers into the viewed vector  $sm_{RS}$  in the spacecraft fixed reference frame  $R_s$ . (see [17] §4) This routines takes into account the scanning geometry of the instrument.

**intposvel** interpolates (a 3 order polynomial interpolation) the satellite position and (relative) velocity in Greenwich reference frame in the SATPOS file, for a given pixel time. This time is included into a  $[t_2, t_3]$  interval of which position and velocity are referenced into SATPOS. These 2 points are used as reference for the polynomial coefficients calculation. The orbit number is also determined for a given pixel. For ATOVS sounders, satellite position is recalculated for every pixel of each scan line. On the contrary, for AVHRR image data (HRPT, GAC), position is computed only for each scan line (assuming that the scanning of a line is instantaneous compared to the satellite velocity).

**snagre** calculates the conversion matrix between Earth fixed Greenwich reference frame  $R_g$  and nominal attitude reference frame  $R_a$ .

**earthpix** calculates the cartesian coordinates  $sm_{RG}$  in the Greenwich reference frame of the viewed pixel  $sm_{RS}$  detailed explanation is given in [17] § 5.

**cartlalo** converts cartesian coordinates  $sm_{RS}$  into latitude-longitude on the earth surface (i.e. altitude = 0)

**zenazi** calculates the zenith angle, azimuth angle and distance of the spacecraft from the viewed point on the earth surface detailed explanation is given in [17] § 6.

**sungrw** calculates the sun position in Greenwich reference frame

**zenazi** calculates the zenith angle, azimuth angle and distance of the sun from the viewed point on the earth surface.

After **nav\_1blin** sets bit flags for variables of the level 1B file, does the conversions for the level 1B units. It computes the satellite altitude (in  $km \cdot 10$ ) by calling **cartgeog** that converts (with iterations) cartesian coordinates ( $R_g$ ) into geographic coordinates (lat/lon/alt). Satellite altitude is determined from the last computed position.

### TASK 3: RESULTS UPDATING

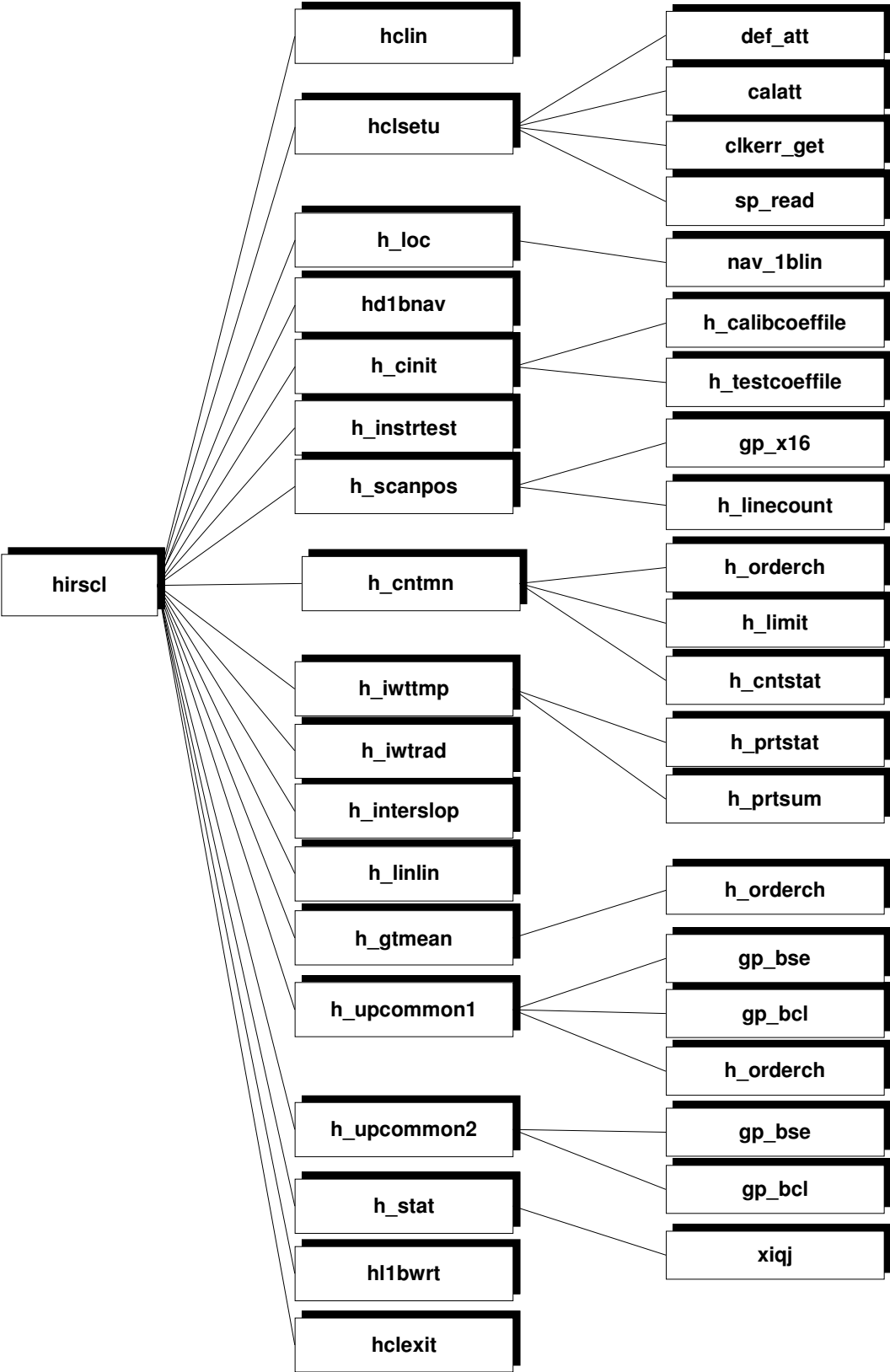
**h\_loc** updates navigation parameters and quality controls within the level 1b file.

**hd1bnav** updates navigation parameters within the level 1b file.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### **4.1.10. HIRS calibration modules (first algorithm): HIRSCL script and HIRSCL.EXE**

(See also reference manual pages: *libhrscal.3*)



**Figure 4-9 : Flow chart on the HIRSCL module components.**

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

To simplify the diagram, the calls to subroutines of the **libf7ml** library have not been written

This task requires HIRS level 1a, *calcoef.dat* and *testcoef.dat* resource files.

#### TASK 1: INITIALISATION

The user chooses his input options (script **hirscl** and **hclin**).

The main program is **hirscl** that calls many routines:

**hclin** reads the input options (tests some options coherence) and stores them into a table.

**hclsetu** opens the log/debug file *hirscl.log* if requested. It opens and reads the HIRS level 1a file (the data are ranged in commons *hrs1bhd* (include *hrs1bhd.h*, header), *hrs1bdts* (include *hrs1bdts*, data)).

**h\_cinit** identifies the satellite. Then it calls **h\_calibcoeffile** to open, read and close the *calcoef.dat* file containing the useful satellite specific parameters for calibration. The data are ranged in common *hrs\_clcf* (include *cinit.h*). **h\_cinit** by calling **h\_testcoeffile**, opens, reads and closes the *testcoef.dat* file containing useful values and parameters for tests. The data are ranged in the common *hrs\_tstcf* (include *cinit.h*). **h\_cinit** opens the statistic file (if requested). A control quality parameter array is initialised and will be modified by the tests performed during the calibration processing. The satellite is identified.

**h\_instrtest** checks the instrument status to define which scan line are usable (the first and the last usable lines). The control quality parameters array is updated.

#### TASK 2: CALIBRATION COEFFICIENTS CALCULATION

The result of this task is a calibration coefficient array (*calibcoef*) for each sounding channel and each scan line. **HIRSCL** calls many routines:

**h\_scanpos**, for each scan line of each channel, checks the 56 encoder positions (quality bit 31 is checked), and keeps the numbers of lines of the calibration cycles (space lines array: *splintab* and internal warm target array: *iwtlintab*). **h\_scanpos** checks if the calibration cycle is full or not and sets up the variable *calib* (number of calibration cycle full). If *calib* equals zero, processing goes directly to the task 3.

**h\_cntmn**, for each calibration cycle of the orbit (for each space and internal warm target lines registered during an orbit) and for each channel, filters numerical counts (CN) and computes the CN mean. Those values are stored in arrays : *spcntmn* (space) and *iwtcntmn* (internal warm target).

**h\_iwttmp**, for each calibration cycle, calculates the internal warm target (IWT) temperature. It computes the mean of a sample of PRT reading from the internal warm target scan line and from a specified number of scan lines before the IWT scan line and another specified number of scan lines after the IWT scan line. **h\_iwttmp** tests the difference between the maximum PRT readings value and the minimum one which must be inferior to a limit before being used in the mean calculation. PRT readings means are converted to temperatures. The final IWT temperature is computed by averaging the temperature from the 4 individual active PRTs (array *iwttmp*). The quality control parameters array is updated.

**h\_iwtrad** converts the IWT temperature (array *iwtrad*) (using the Planck function, applying bands correction) for each channel and each calibration cycle.

**h\_interslop** for each calibration cycle and each channel, computes gain G and offset I (residual radiance equivalent to the space background noise viewed through the instrument channel) (array *calibcoef0*). The coefficients of the visible channel are not measured in flight. A third coefficient (order 2) is also designed for the calculation in addition to G and I. It is equal to zero for the moment and so it is not yet used.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**h\_linlin** for each channel and each Earth viewing scan line, computes the (G,I) pairs (array *calibcoef*) by linear interpolation between 2 pairs of coefficients (G,I) calculated for 2 consecutive calibration cycles. For Earth viewing registered before the first calibration cycle there is no interpolation, coefficients of the first calibration cycle are directly applied. For Earth viewing registered after the last calibration cycle there is no interpolation, coefficients of the last calibration cycle are directly applied. The quality control parameters array is updated.

**h\_gtmean** for each channel, computes the mean (array *calibcoefmn*) and the standard deviation (array *calibcoefstd*) of the coefficients (G,I). The header is updated in the level1b file.

### TASK 3: RESULTS UPDATING

The result of this task is an update of calibration coefficients, and quality control parameters in the HIRS level 1b resource file. According to input options, statistics results are stored into the file *monhirs.txt*, and the log/debug file is updated.

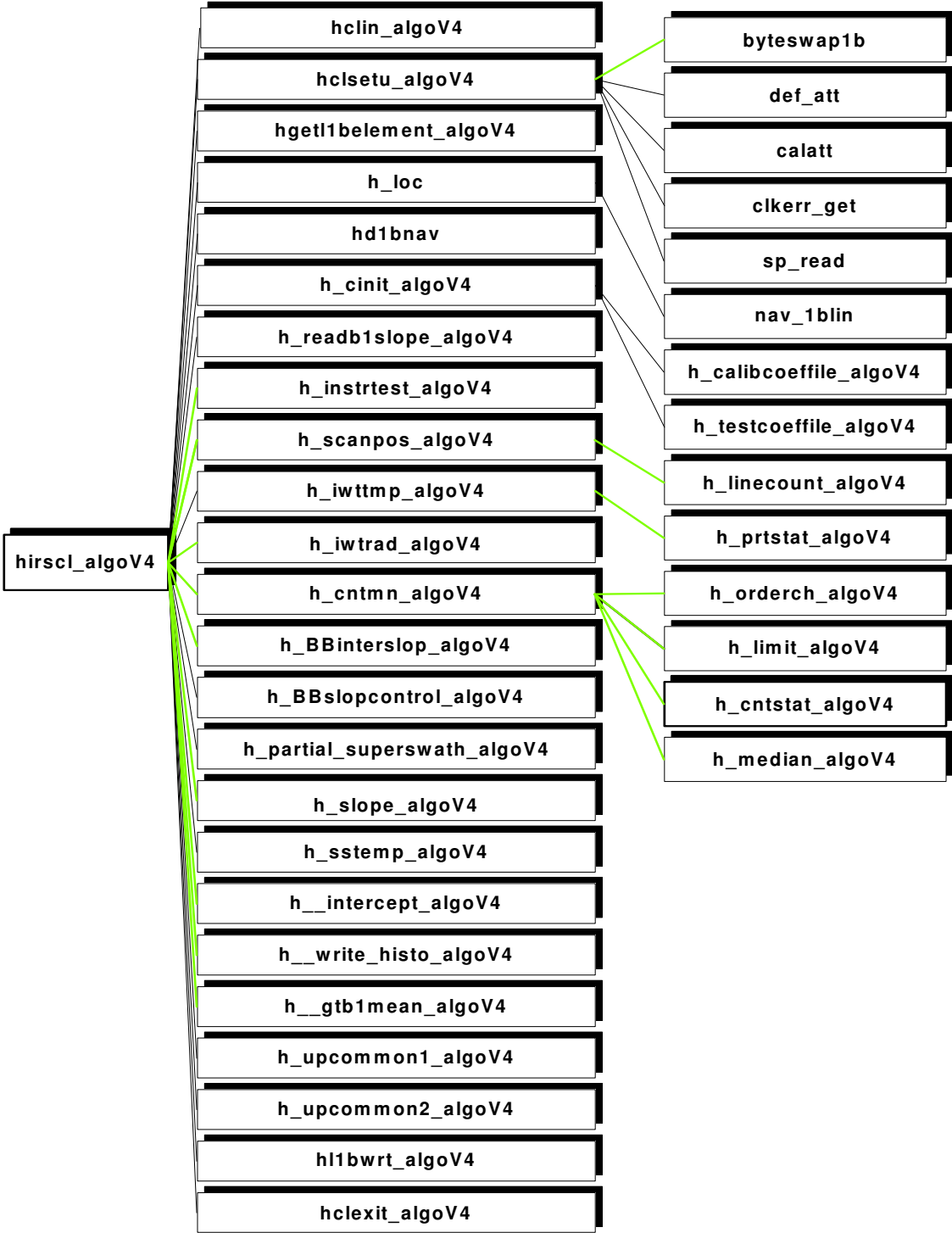
**h\_upcommon1** or **h\_upcommon2** finish updating the parameters in the commons *hrs1bhd*, and *hrs1bdts* (**h\_upcommon2** is called when there is no calibration).

**h\_stat** computes final statistic of the HIRS calibration and writes the results into the statistic file, and then closes the file (according to input options).

**hl1bwrt** updates header and data in the HIRS level1b file.

**hclexit** close the log/debug file and the HIRS level1b file.

#### **4.1.11. HIRS calibration modules ( algorithm version 4): HCALCB1\_ALGOV4 script and HCALCB1\_ALGOV4.EXE, HIRSCL\_ALGOV4 script and HIRSCL\_ALGOV4.EXE**



**Figure 4-10 : Flow chart on the HIRSCl\_ALGOV4 module components.**

To simplify the diagram, the calls to subroutines or functions of the **libf7ml**, **libsatid**, **libf7gp**, **libf7tp** libraries have not been written

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

This version of the HIRS calibration doesn't work for the pre-NOAA-K satellites.

`hirscl_algoV4` requires HIRS level 1a, `calcoef_algoV4.dat`, `testcoef_algoV4.dat` and `hirs_biaslope.txt` resource files.

#### TASK 1: CREATE THE HIRS BIASLOPE.TXT FILE

The user chooses a reference date/time, a number of hours and the B1 coefficients and the average slopes will be computed using data of the period defined by [ (the reference date/time – the number of hours) – (the reference date/time)]. The reference date/time is the input arguments of the script `hcalcb1_algoV4`. Note that `AAPP_RUN` calls the script `hcalcb1_algoV4` with the date/time of the current orbit. The number of hours is defined in `ATOVS_ENV`.

An other option is defined by the user in `ATOVS_ENV: HCALIB_BIASLOP_FLAG`

(=0 if the user doesn't want to have the time taken into account to define the period; =1 if the user want to have the time (hours/minutes) taken into account to define the period).

The main program `hcalcb1_algoV4.exe` requires a hirs historic file. If the file doesn't exist, `hcalcb1_algoV4` creates it, it will be empty.

`hcalcb1_algoV4.exe` calls the routine `h_calcb1_algoV4` that reads the `hirs_historic` file (call to the routine `h_read_histo_algoV4`) and does the computations (call to the routines `moy_rms`, `reglin`).

`hcalcb1_algoV4.exe` manages the openings/writings/closings of the different files.

#### TASK 2: INITIALISATION

The script `hirscl_algoV4` must run with the argument “-c” for doing the calibration task (see `AAPP_RUN`).

The main program is `hirscl_algoV4.exe` that calls many routines:

`hclin_algoV4` reads the input options (tests some options coherence) and stores them into a table.

`hclsetu_algoV4` opens the log/debug file `hirscl.log` if requested. It opens and reads the HIRS level 1a file.

`hgetl1belement_algoV4` gets the elements of the HIRS level 1b commons that are useful for the calibration task.

`h_cinit_algoV4` identifies the satellite. Then it calls `h_calibcoefficient_algoV4` to open, read and close the `calcoef_algoV4.dat` file containing the useful satellite specific parameters for calibration. By calling `h_testcoefficient_algoV4`, it opens, reads and closes the `testcoef_algoV4.dat` file containing useful values and parameters for tests. It initializes the elements in the includes that will be updated during the calibration task.

`h_readb1slope_algoV4` opens/closes and reads the 'b1/average slope' file to get the b1 values and the average slopes that will be used in the process.

`h_instrtest_algoV4` checks the instrument status to define which scan line are usable (the first and the last usable lines). The control quality array of scan lines is updated.

#### TASK 3: CALIBRATION COEFFICIENTS CALCULATION

The result of this task is a calibration coefficient array (`calibcoef`) for each sounding channel and each scan line. `hirscl_algoV4` calls many routines:

`h_scanpos_algoV4`, for each scan line, checks the quality bit 31, the 56 encoder positions and the line counts. It keeps the numbers of lines of the calibration cycles (space lines array: `splintab` and internal warm target array: `iwtlintab`). It also checks if the calibration cycle is full or not, checks if one calibration cycle and the previous one are well separated by 40 scan lines, and sets up the



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

variable *calib* (number of calibration cycle full). If *calib* equals zero, processing goes directly to the task 3.

**h\_iwttmp\_algoV4**, for each calibration cycle, calculates the internal warm target (IWT) temperature: For each individual active PRT, it gets a sample of PRT readings from the internal warm target scan line and from a specified number of scan lines before the IWT scan line and another specified number of scan lines after the IWT scan line. **h\_iwttmp\_algoV4** tests the PRT readings before being used in the mean calculation. The mean of the PRT readings is converted to temperature. The final IWT temperature is computed by averaging the temperatures from the 4 (5 for NOAA-N) individual active PRTs (array *iwttmp*). The quality control parameters array is updated.

**h\_iwtrad\_algoV4** converts the IWT temperatures into radiances (array *iwtrad*) (using the Planck function, applying bands correction) for each channel and each calibration cycle.

**h\_cntmn\_algoV4**, for each calibration cycle of the orbit (for each space and internal warm target lines registered during an orbit) and for each channel, filters numerical counts (CN) and computes the CN mean. Those values are stored in arrays : *spcntmn* (space) and *iwtcntmn* (internal warm target).

**h\_BBinterslop** for each calibration cycle and each channel, computes the Black Body (BB) (or raw) slope (auto coefficient 1), the BB (or raw) intercept (auto coefficient 0) and third coefficient (auto coefficient 2) that is equal at zero for the moment (array *calibcoef0*). The coefficients of the visible channel are not measured in flight.

**h\_BBslopcontrol** controls the quality of the Bbslopes.

**h\_partial\_superswath\_algoV4** determines the calib cycles which will be involved in the calculation of average slope for each superswath or partial superswath.

**h\_slope\_algoV4** computes the calibration slopes for each channel and for each Earth view scan line.

**h\_sttemp\_algoV4** computes the Second Telescope Temperature for all lines.

**h\_intercept\_algoV4** computes the intercept for each channel and for each Earth view line.

#### TASK 4: RESULTS UPDATING

The result of this task is an update of calibration coefficients, and quality control parameters in the HIRS level 1b resource file.

**h\_write\_histo\_algoV4** stores calibration information of all calibration cycles in the *hirs\_historic* ASCII file.

**h\_gtb1mean\_algoV4** computes the means and the standard deviations of the b1 coefficients of all the lines.

**h\_upcommon1\_algoV4** or **h\_upcommon2\_algoV4** finish updating the parameters in the commons *hrs1bhd*, and *hrs1bdts* (**h\_upcommon2\_algoV4** is called when there is only one or zero calibration cycle).

**h11bwrt\_algoV4** updates header and data in the HIRS level1b file.

**h\_clexit\_algoV4** closes the log/debug file and the HIRS level1b file.

### 4.1.12. MSU calibration modules: MSUCL script and MSUCL.EXE

(See also reference manual pages: *libmsucal.3*)

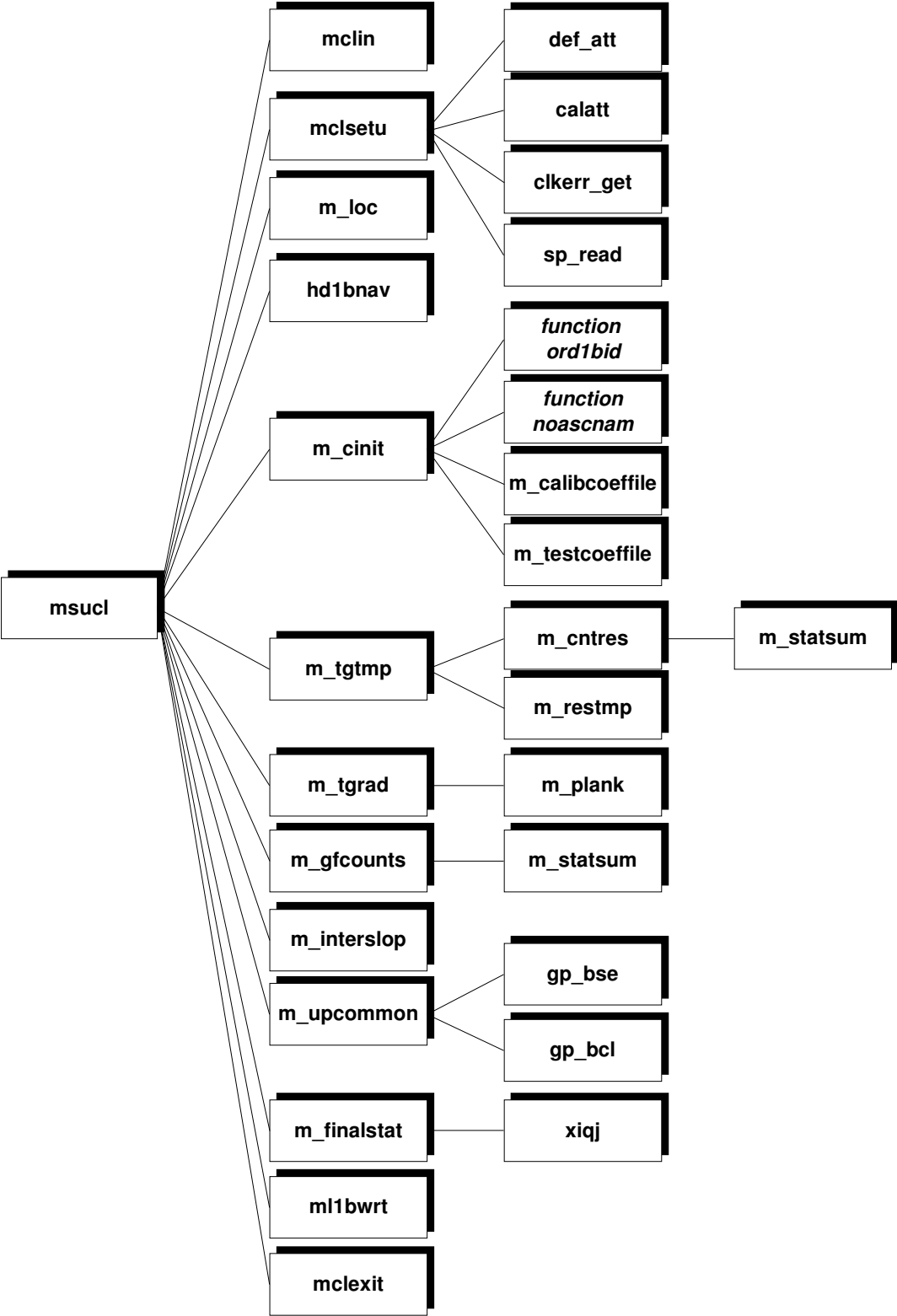


Figure 4-11 : Flow chart on the MSUCL module components.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

To simplify the diagram, the calls to subroutines of the **libf7ml** library have not been written

This task requires MSU level 1a, *calcoef.dat* and *testcoef.dat* resource files .

#### TASK 1: INITIALISATION

The user chooses his input options (script **msucl** and **mclin**).

The main program is **MSUCL** that calls many routines:

**mclin** reads the input options (tests some options coherence) and stores them in a table.

**mclsetu** opens the log/debug file *muscl.log* if requested. It opens and reads the MSU level 1a file *msu1b* (the data are ranged in commons *msu1bhd* (header), *msu1bdts*(data)).

**m\_cinit** opens, reads and closes (**m\_calibcoefffile**) the *calcoef.dat* file containing the useful satellite specific parameters for calibration. The data are ranged in common *msu\_clcf* (include *mcinit.h*). **m\_cinit** opens, reads and closes (**m\_testcoefffile**) the *testcoef.dat* file containing useful values and parameters for tests. The data are ranged in the common *msu\_tstcf* (include *mcinit.h*). **m\_cinit** opens the statistic file (if requested). Two control quality parameters arrays are updated.

#### TASK 2: CALIBRATION COEFFICIENTS CALCULATION

The result of this task is two calibration coefficient arrays (slope, intercept) for each sounding channel. **msucl** calls many routines:

**m\_tgtmp**, for each scan line, computes target 1 temperature and target 2 temperature, which are derived respectively from PRT 1A, 1B counts. Target 1 is viewed by channels 1 and 2, target 2 is viewed by channels 3 and 4.

To convert PRT count to temperature requires two steps :

- Convert count to resistance (call **m\_entres**)
- Convert resistance to temperature (call **m\_restmp**)

Conversion parameters are tabulated in the *calcoef.dat* file. **m\_tgtmp** tests the low (*tcallo*) and high (*tcalli*) values of the electronic reference points and sets a flag if values are out of limits. It tests the temperature calculated with a reference. If the difference is higher than a threshold value, then reference temperature is kept. Final temperature of each IWT is the mean of the two associated PRT temperatures (arrays *tg1* and *tg2*). Two control quality parameters arrays are updated.

**m\_tgrad**, for each scan line, converts the target 1 temperature into radiance for MSU channels 1 and 2 (array *tgrad*) (apply Planck function). Same for the target 2 temperature but for the MSU channels 3 and 4.

**m\_gfcounts**, for each scan line of each channel, applies the non linearity coefficients on the space view output counts and the target output counts. **m\_gfcounts** applies a filter (abs(count - averaged count) compared to ( 2\*standard deviation)) to eliminate counts out of limits. Different parameters are stored in the statistic file (if requested). Array 2 of quality control parameters is updated.

**m\_interslop**, for each calibration cycle and each channel, computes the gain G and the offset I (residual radiance equivalent to the space background noise viewed through the instrument channel). The coefficients used afterwards (arrays slope and intercept), are averaged pairs (G,I) coming from the calculation of the mean of the (G,I) values associated to each scan line and each channel during an orbit.

<p><b>NWP SAF</b></p>	<p><b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b></p>	<p>Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019</p>
-----------------------	---	--

TASK 3: RESULTS UPDATING

The result of this task is an update of the calibration coefficients and the quality control parameters in the MSU level 1b resource file. According to input options, statistics results are stored into the file *monmsu.txt*, and a log/debug file is updated.

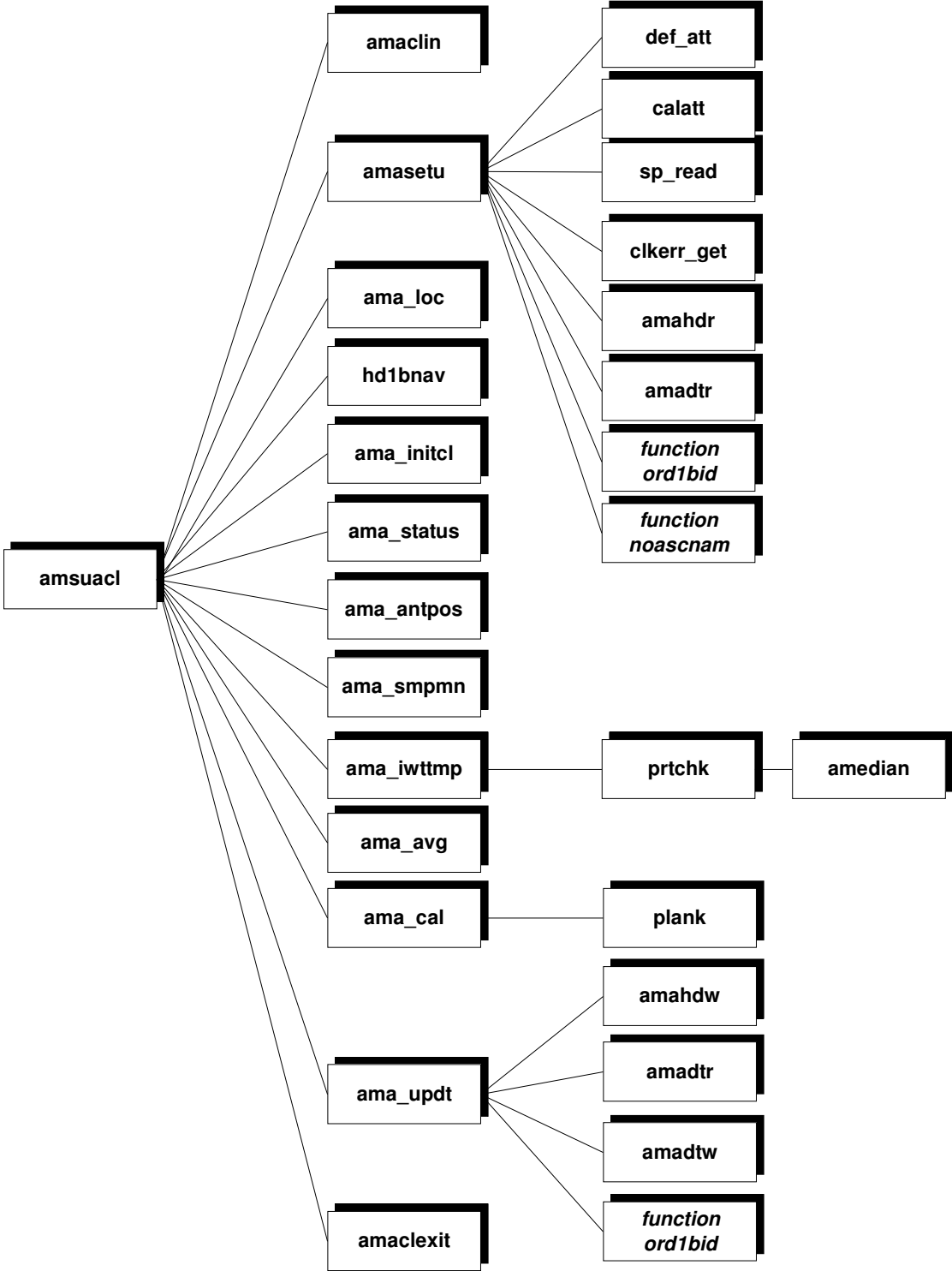
**m\_upcommon** updates the commons *msu1bhd* and *msu1bdts*.

**m\_finalstat** computes the final statistic of the MSU calibration and writes the results into the statistic file, and then closes the file (according to input options).

**m11bwrt** updates header and data in the MSU level 1b file.

**mclexit** close the log/debug file and the MSU level 1b file.

**4.1.13. AMSU-A calibration modules : AMSUACL script and AMSUACL.EXE.**



**Figure 4-12 : Flow chart on the AMSUCL module components.**

To simplify the diagram, the calls to subroutines of the **libf7ml** library have not been written

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

This task requires the AMSU-A level 1a, *amsua\_clcoefs.dat* and *amsua\_clparams.dat* resource files.

#### TASK 1: INITIALISATION

The user chooses his input options (script **amsuac1** and **amaclin**).

The main program is **AMSUACL** that calls many routines:

**amaclin** reads the input options and stores them in a table.

**amasetu** opens the log/debug file *amsuac1.log* and the statistic file *monamsua.txt* (if requested). It opens and reads the AMSUA level 1a file (the data are ranged in commons *ama\_1bhd* (include *ama1b.h*, header), and scan (include *amascn.h*, data). **amasetu** checks the satellite Id and data, and then set some control flags.

**ama\_initcl** opens, reads and closes the *amsua\_clparams.dat* file containing the useful parameters for calibration. The data are arranged in the common *ama\_clcoef* (include *ama\_cinit.h*). **ama\_initcl** opens, reads and closes the *amsua\_clcoefs.dat* file containing the values of the secondary calibration coefficients. The data are arranged in the common *ama\_tstcf* (include *ama\_cinit.h*). Quality control flags are updated.

**ama\_status** determines if the instrument is OK and sets flags according to the results : checks scan lines quality, checks space viewing antenna positions, checks calibration counts and channels. If not OK, calibration coefficients are not computed for the bad scan line, but will be replaced by secondary coefficients (*amsua\_clcoefs.dat*).

#### TASK 2: CALIBRATION COEFFICIENTS CALCULATION

The results of this task is the primary calibration coefficient for each sounding channel.

**ama\_antpos** checks if the antenna pointing of the AMSU Earth view is not outside of the specified threshold.

**ama\_smpmn** gets CN samples and computes the mean. For each channel and each scan line, these averaged values are stored in the arrays *spmean* (space) and *itmean* (IWT). Quality control flags are updated.

**ama\_iwttmp**, for each scan line, converts PRT counts to temperature for IWT and instruments. Final temperature of each IWT is a weighted average of the temperatures extracted from their associated PRT. Arrays of averaged temperatures *targ\_temp* and *inst\_temp* are filled. Quality control flags are updated.

**ama\_avg** computes mean counts for space and IWT. Averaging is performed on several consecutive lines for each channel. These mean values fill arrays *spavg* (space) and *itavg* (IWT). Quality control flags are updated.

**ama\_cal**, for each line and each channel, computes calibration coefficients from space and IWT data: performs temperature/radiance conversion, deduces primary calibration coefficients (*a0,a1,a2*). Primary and secondary coefficients (*u0,u1,u2*) are stored in the commons *ama\_1bhd* and *scan*. The quality control flags are updated.

#### TASK 3 : RESULTS UPDATING

The result of this task is an update of calibration coefficients and quality control parameters, in the AMSU-A level 1b resource file.

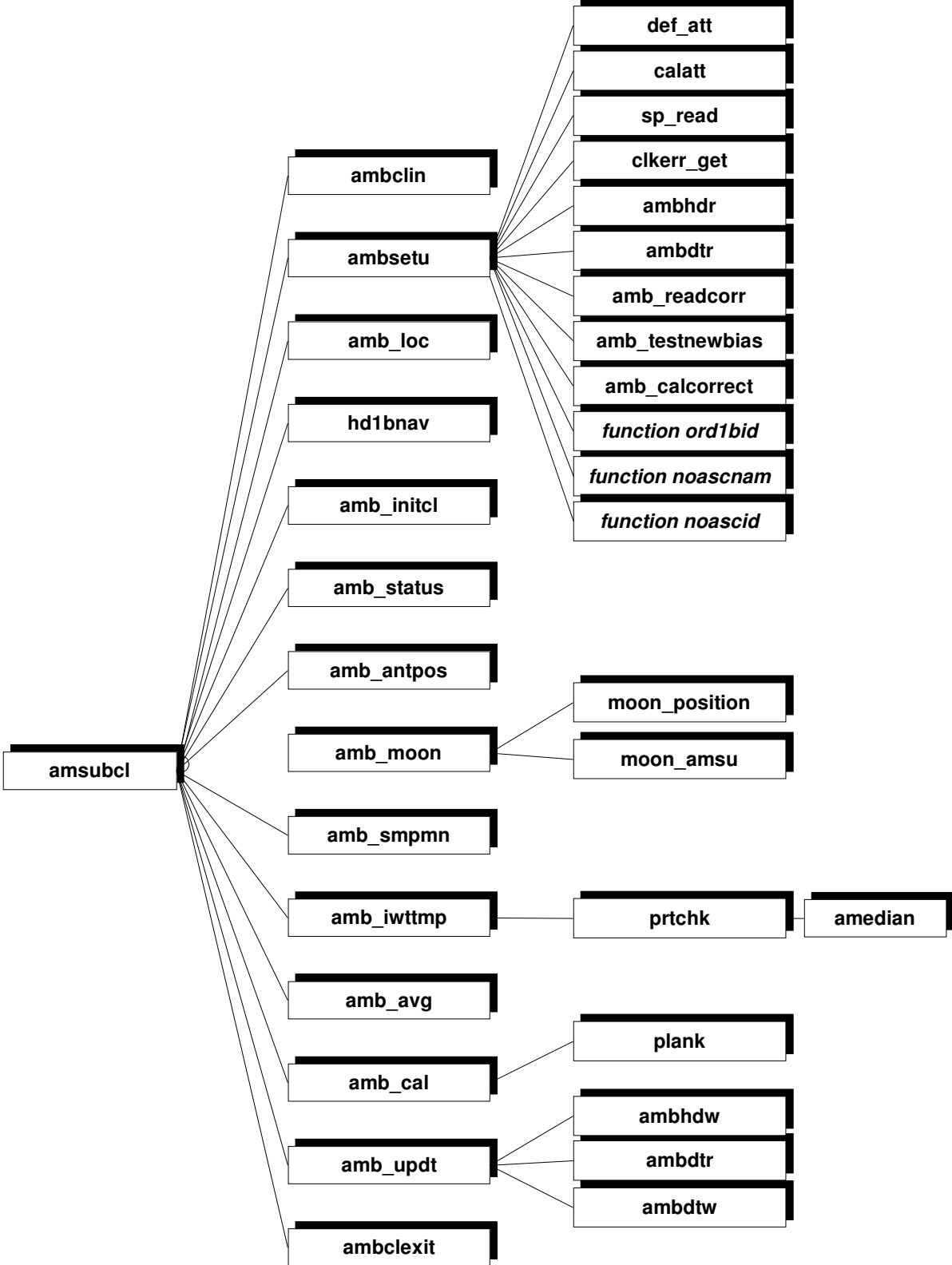
According to input options, statistics results are stored into the stat file and a log/debug file is updated.

**ama\_updt** updates header and data in the AMSU-A level 1b.

**amaclexit** closes the log/debug file and the AMSU-A level 1b file

.

**4.1.14. AMSU-B calibration modules: AMSUBCL script and AMSUBCL.EXE.**



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**Figure 4-13 : Flow chart on the AMSUBCL module components.**

To simplify the diagram, the calls to subroutines of the **libf7ml** library have not been written

This task requires the AMSU-B level 1a, *amsub\_clcoefs.dat*, *amsub\_clparams.dat* and *amsub\_bias.dat* resource files .

#### TASK 1: INITIALISATION

The user chooses his input options (script **amsubcl** and **ambclin**).

The main program is **AMSUBCL** that calls many routines:

**ambclin** reads the input options and stores them in a table.

**ambsetu** opens the log/debug file *amsubcl.log* and the statistic file *monamsub.txt* (if requested). It opens and reads the AMSU-B level 1a file (the data are ranged in commons *amb\_1bhd* (include *amb1b.h*, header), and scan (include *ambscn.h*, data). **ambsetu** calls **amb\_readcorr** to read bias correction tables and stores in level 1b header. It calls **amb\_testnewbias** to detect presence of AMSU-B anomalous bias due to moding of STX-1 transmitter. Then, it calls **amb\_calcorrect** to correct AMSU-B space and target counts for bias errors. **ambsetu** checks the satellite Id and data, and then sets some control flags.

**amb\_initcl** opens, reads and closes the *amsub\_clparams.dat* file containing the useful parameters for calibration. The data are arranged in the common *amb\_clcoef* (include *amb\_cinit.h*). **amb\_initcl** opens, reads and closes the *amsub\_clcoefs.dat* file containing the values of the secondary calibration coefficients. The data are arranged in the common *amb\_tstcf* (include *amb\_cinit.h*). Quality control flags are updated.

**amb\_status** determines if the instrument is OK and sets flags according to the results : checks scan lines quality, checks space viewing antenna positions, checks calibration counts and channels. If not OK, calibration coefficients are not computed for the bad scan line, but will be replaced by secondary coefficients (*amsub\_clcoefs.dat*).

#### TASK 2 : CALIBRATION COEFFICIENTS CALCULATION

The result of this task is the primary calibration coefficient for each sounding channel.

**amb\_antpos** checks if the antenna pointing of the AMSU Earth view is not outside of the specified threshold.

**amb\_moon** calculates the angles between the Moon and the AMSU-B space views for all scans, based on astronomical formulae.

**amb\_smpmn** gets calibration samples and computes the mean. If any of the space samples are within a pre-defined angle to the Moon, they are excluded from the mean. For each channel and each scan line, these averaged values are stored in the arrays *spmean* (space) and *itmean* (IWT). Quality control flags are updated.

**amb\_iwttmp**, for each scan line, converts PRT counts to temperature for IWT and instruments. Final temperature of each IWT is a weighted average of the temperatures extracted from their associated PRT. Arrays of averaged temperatures *targ\_temp* and *inst\_temp* are filled. Quality control flags are updated.

**amb\_avg** computes mean counts for space and IWT. Averaging is performed on several consecutive lines for each channel. These mean values fill arrays *spavg* (space) and *itavg* (IWT). Quality control flags are updated.

**amb\_cal**, for each line and each channel, computes calibration coefficients from space and IWT data: performs temperature/radiance conversion, deduces primary calibration coefficients (*a0,a1,a2*). Primary and secondary coefficients (*u0,u1,u2*) are stored in the commons *amb\_1bhd* and *scan*. The quality control flags are updated.



<p><b>NWP SAF</b></p>	<p><b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b></p>	<p>Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019</p>
-----------------------	---	--

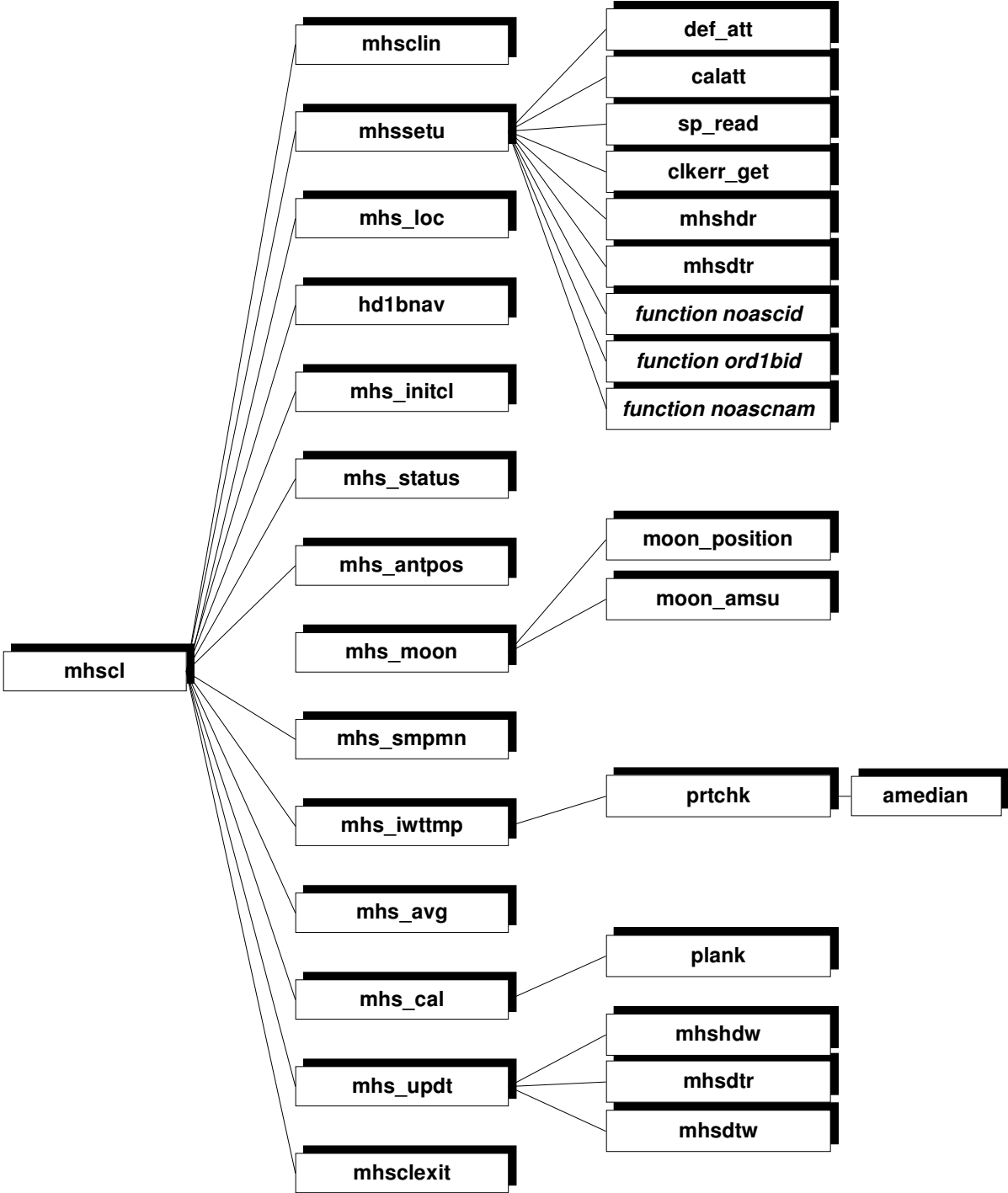
TASK 3 : RESULTS UPDATING

The result of this task is an update of calibration coefficients and quality control parameters, in the AMSU-B level 1b resource file. According to input options, statistics results are stored into the stat file and a log/debug file is updated.

**amb\_updt** updates header and data in the AMSU-B level 1b.

**ambclexit** closes the log/debug file and the AMSU-B level 1b file.

**4.1.15. MHS calibration modules: MHSCL script and MHSCL.EXE.**



**Figure 4-14 : Flow chart on the AMSUBCL and MHSCL module components.**

To simplify the diagram, the calls to subroutines of the **libf7ml** library have not been written

This task requires the MHS level 1a, *mhs\_clcoefs.dat* and, *mhs\_clparams.dat* resource files.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

### TASK 1: INITIALISATION

The user chooses his input options (script **mhscl** and **mhsclin**).

The main program is **MHSCL** that calls many routines:

**mhsclin** reads the input options and stores them in a table.

**mhssetu** opens the log/debug file *mhscl.log* and the statistic file *monmhs.txt* (if requested). It opens and reads the MHS level 1a file (the data are ranged in commons *mhs\_1bhd* (include *mhs1b.h*, header), and scan (include *mhssc.h*, data). **mhssetu** checks the satellite Id and data, and then sets some control flags.

**mhs\_initcl** opens, reads and closes the *mhs\_clparams.dat* file containing the useful parameters for calibration. The data are arranged in the common *mhs\_clcoef* (include *mhs\_cinit.h*). **mhs\_initcl** opens, reads and closes the *mhs\_clcoefs.dat* file containing the values of the secondary calibration coefficients. The data are arranged in the common *mhs\_tstef* (include *mhs\_cinit.h*). Quality control flags are updated.

**mhs\_status** determines if the instrument is OK and sets flags according to the results : checks scan lines quality, checks space viewing antenna positions, checks calibration counts and channels. If not OK, calibration coefficients are not computed for the bad scan line.

### TASK 2 : CALIBRATION COEFFICIENTS CALCULATION

The result of this task is the primary calibration coefficient for each sounding channel.

**mhs\_antpos** checks if the antenna pointing of the AMSU Earth view is not outside of the specified threshold.

**mhs\_moon** calculates the angles between the Moon and the MHS space views for all scans, based on astronomical formulae.

**mhs\_smpmn** gets calibration samples and computes the mean. If any of the space samples are within a pre-defined angle to the Moon, they are excluded from the mean. For each channel and each scan line, these averaged values are stored in the arrays *spmean* (space) and *itmean* (IWT). Quality control flags are updated.

**mhs\_iwttmp**, for each scan line, converts PRT counts to temperature for IWT and instruments. Final temperature of each IWT is a weighted average of the temperatures extracted from their associated PRT. Arrays of averaged temperatures *targ\_temp* and *inst\_temp* are filled. Quality control flags are updated.

**mhs\_avg** computes mean counts for space and IWT. Averaging is performed on several consecutive lines for each channel. These mean values fill arrays *spavg* (space) and *itavg* (IWT). Quality control flags are updated.

**mhs\_cal**, for each line and each channel, computes calibration coefficients from space and IWT data: performs temperature/radiance conversion, deduces primary calibration coefficients (*a0,a1,a2*). Primary and secondary coefficients (*u0,u1,u2*) are stored in the commons *mhs\_1bhd* and *scan*. The quality control flags are updated.

### TASK 3 : RESULTS UPDATING

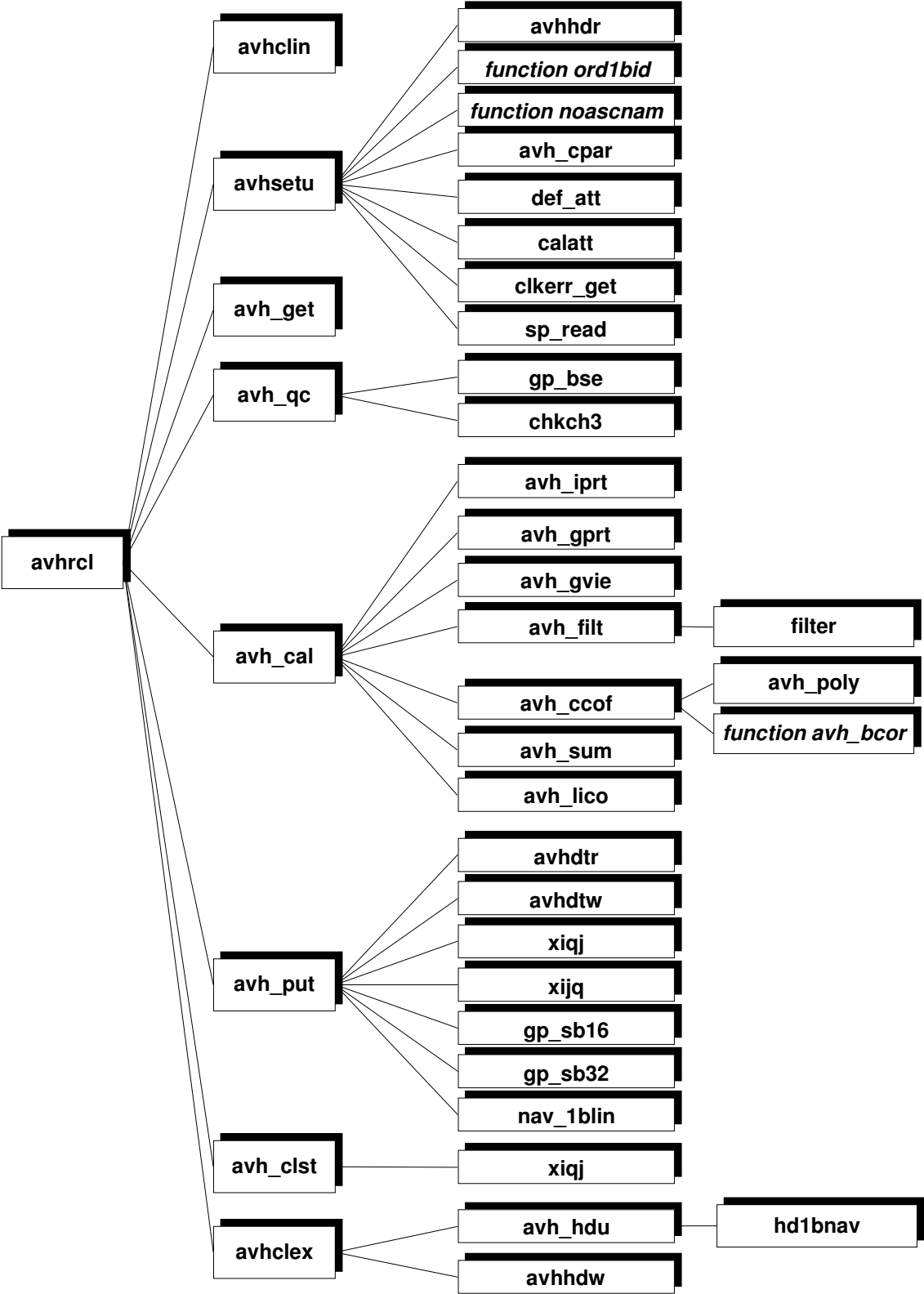
The result of this task is an update of calibration coefficients and quality control parameters, in the MHS level 1b resource file. According to input options, statistics results are stored into the stat file and a log/debug file is updated.

**mhs\_updt** updates header and data in the MHS level 1b.

**mhsclexit** closes the log/debug file and the MHS level 1b file.

**4.1.16. AVHRR calibration module: AVHRCL script and AVHRCL.EXE.**

(See also the reference manual pages *libavhcal.3*)



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**Figure 4-15 : Flow chart on the AVHRCL module components.**

To simplify the diagram, the calls to subroutines of the **libf7ml** library have not been written

This task requires the AVHRR level 1a and *avhcal.txt* resource files.

#### TASK 1: INITIALISATION

The user chooses his input configuration (options). A statistic file is opened. Quality control flags are set and will be modified all along the program performance.

**avhclin** reads the input options and stores them in a table.

**avhsetu** opens the statistic file *monavhr.txt*, opens and reads the AVHRR level 1a file (the data are ranged in *avh1b.h*). Satellite Id is set. **avhsetu** opens, reads and closes the file *avhcal.txt* containing the useful parameters for satellite specific calibration.

**avh\_get** reads the AVHRR level 1a file (only the part needed by calibration) and stores the data into memory.

**avh\_qc** checks the quality of each AVHRR scan line from the file and flags the lines having bad scan numbers inside the level 1b file. Bad line numbers are corrected. **avh\_qc** gets the first calibration sequence from the data.

#### TASK 2: CALIBRATION COEFFICIENTS CALCULATION

The result of this task is the calibration coefficients for each channel and each scan line.

**avh\_cal** manages the main loop for AVHRR calibration :

- initialises thermistors PRT counts arrays (**avh\_iprt**).
- for each calibration cycle, fills the count arrays for the 4 PRTs.
- fills the IWT count arrays and the space count arrays (**avh\_gvie**).
- pass Gross and Sigma filters to eliminate noisy counts (**avh\_filt**).
- computes coefficients for each AVHRR calibration cycle (converts mean PRT counts to mean IWT temperature, computes target radiance and deduces (G,I) coefficients (**avh\_ccof**) and calibration coefficients (*k1,k2,k3*), and linearises (**avh\_lico**) the coefficients for each scan line.

#### TASK 3: RESULTS UPDATING

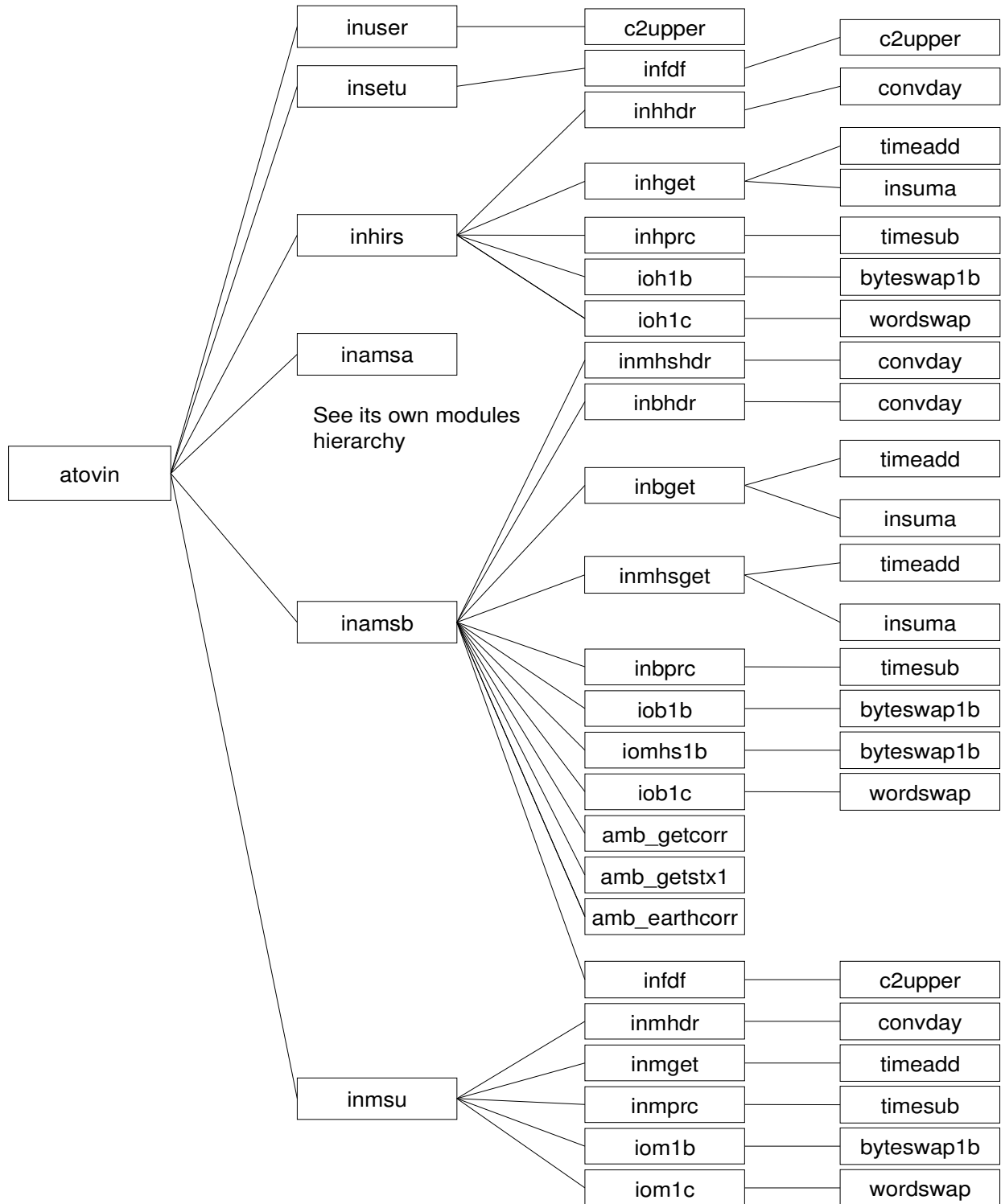
The result of this task is an update of the calibration coefficients in the AVHRR level 1a resource file. Statistics results are stored into the file *monavhr.txt*.

**avh\_put** updates data in the AVHRR level 1b file.

**avh\_clst** finishes the statistic calculation relative to calibration, writes the results and closes the statistics file.

**avhclx** updates header and closes the AVHRR level 1b file.

### **4.1.17. ATOVS sounders calibration: ATOVIN script and ATOVIN.EXE**



**Figure 4-16 : ATOVIN module hierarchy**

To simplify the diagram, calls to the **errorreport** subroutine and **numdays** function have not been written

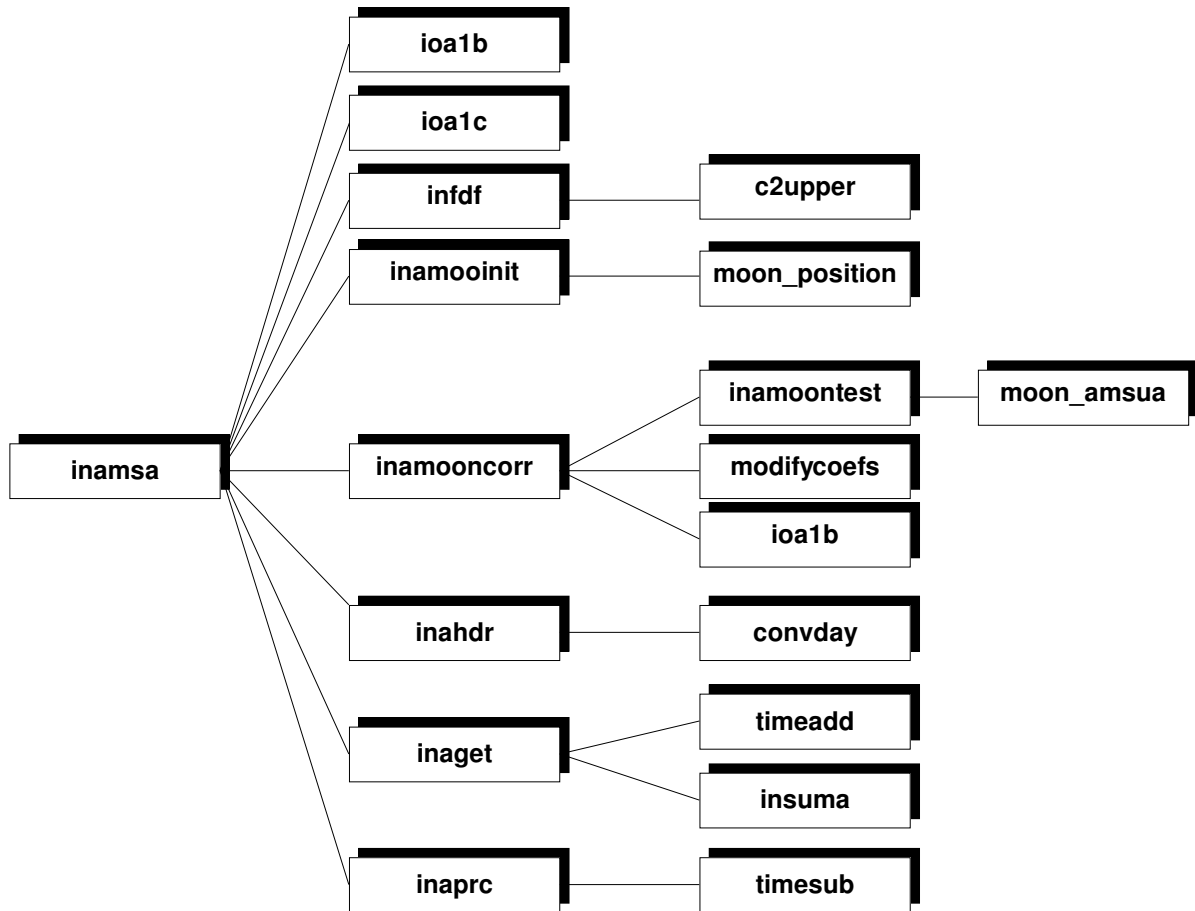


Figure 4-17 : INAMSA module hierarchy

This task requires the level 1b files of each instrument, together with *fdf.dat* and *stx1\_mar99corr.dat* resource files.

It applies calibration coefficients (computed by **atovcl**) to output counts to produce radiances. Then it performs radiance conversion to brightness temperature (for each channel). This results in one file for each instrument containing navigated data converted to brightness temperature. Those files represent the level 1c of the processing chain.

#### TASK 1: INITIALISATION

This task performs all the set up operations for the program **atovin**.

The subroutine **inuser** performs the reading of the list of instruments to process from standard input. It performs also the set up of the logical units associated with the instruments data I/O files and the fixed data file (see next chapter).

The subroutine **insetu** performs all the initialisations needed for **atovin** processing. It performs fixed data file reading (**infdf**) and defines bit numbers (convention used in 1b & 1c files is that an integer\*4 word has bits numbered 0-31, with bit 0 being the least significant bit. Some platforms take bit 31 as the low significant bit). Here we explicitly define the order of bits that we use, to keep the code portable.

The subroutine **infdf** reads the following data :

- list of satellite Ids (NESDIS, NOAA & WMO code).

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

- nominal satellite heights & orbit periods.
- AMSU-A & B and MHS antenna efficiencies for antenna corrections [5]
- MHS antenna reflectivity factors for scan-dependent correction (also available for AMSU-B if required)

This task returns to **atovin**: instruments to process, files logical units and initialised variables needed for processing.

#### TASK 2: CALIBRATION OF INSTRUMENT TO PROCESS

This task performs the following functions (data are to be processed one instrument at a time and one scan line at a time):

- it reads Earth-located counts and calibration in level 1B format for each instrument separately (HIRS, AMSU-A, AMSU-B, MHS, MSU).
- it applies the calibration coefficients and converts radiances to brightness temperature.
- it corrects AMSU-A & B and MHS radiances for antenna effects.
- it performs quality control including :
  - to check that the data set increments consistently in time (level 1b data should already have this attribute, and problem detected here indicates a problem with an earlier processing module).
  - to check that the brightness temperatures are within reasonable bounds, substituting missing values if they are not.
- it writes out, for each instruments separately, Earth-located brightness temperatures in level 1c format.

This task is performed by calling the subroutines **inhirs**, **inamsa**, **inamsb** and **inmsu** respectively for level 1b HIRS, AMSU-A, AMSU-B/MHS and MSU data.

In the following part the X depends on the sounder to process (X = h for HIRS, a for AMSU-A, b for AMSU-B, mhs for MHS and m for MSU).

First, it opens level 1b (**ioX1b**) and level 1c files (**ioX1c**), reads level 1b header and sets up level 1c header (**inXhdr**). For MHS, a dedicated subroutine is provided for 1b reading (**iomhs1b**), but the 1c format is shared with AMSU-B so there is no corresponding I/O routine for MHS level 1c. Then, for each scan line read (**ioX1b**), it stores level 1b data into level 1c commons and arrays (**inXget**). It converts counts to radiance and then to brightness temperatures (**inXpre**). Finally it writes the scan line in the level 1c file corresponding to the processed sounder (**ioX1c**).

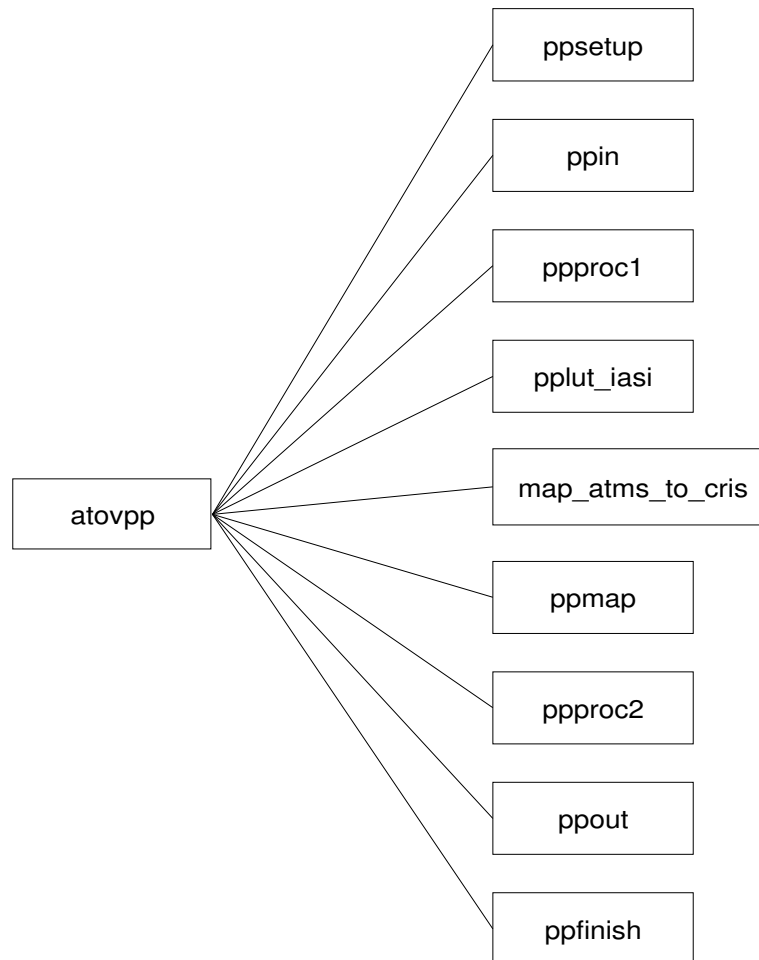
For HIRS, AMSU-A and AMSU-B routine **inXget** calls the subroutine **insuna**, to compute solar zenith and azimuth angles. Additionally, for AMSU-B, the subroutine **inamsb** calls the routines **amb\_getcorr** to read and interpolate bias coefficients, **amb\_getstx1** to read and interpolate antenna corrections, and **amb\_earthcorr** to correct earth-view counts for bias errors. For AMSU-A, the subroutine **inamsa** calls different routines to apply a moon detection/correction. **inamooninit** generates initial fixed values. It calls **moon\_position** that calculates the position of the moon. **inamooncor** determines if the moon is in the AMSU-A ifov (**inamootest**). **modifycoefs** calculates gain and optionally over-write the calibration parameters.

To finish, the level 1c header is completed and written in the level 1c file and the files are closed (**ioX1b** and **ioX1c**)

#### 4.1.18. Mapping of sounders: ATOVPP script and ATOVPP.EXE.



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---



**Figure 4-18 : ATOVPP modules hierarchy**

This task requires the level 1c files of each instrument and several resource files.

The ATOVPP script creates links to the resource files and creates a text file `atovpp.inp` containing the mapping requirements for ATOVPP.EXE. It then checks the `IASI.fdf` file (in `$DIR_IASI_PREPROC`) to see whether a Principal Components analysis has been requested; if so, it creates the necessary binary eigenvectors files (from the supplied gzipped text files) via a call to `IASI_EIGENVECTORS.EXE`.

If ATMS is input, run **ATMS\_BEAMWIDTH** script in order to modify the ATMS file `atms.llc`, according to the required beamwidth specification.

**ATOVPP.EXE** identifies and flags data contaminated by precipitation and maps data of one sounder to the grid of another: e.g. HIRS + AMSU-A + AMSU-B to HIRS grid, AMSU-A + AMSU-B to AMSU-B grid, AMSU-A + MHS to IASI grid, ATMS to CRIS grid. Mapping is the process of calculating a representative value for the data of one instrument (the 'mapping' instrument) at the location of a field of view (fov) of a second instrument (the 'target' instrument). The process of mapping can be considered as three separate steps:

Pre-processing: sets quality flags for mapping fovs (precipitation).

Colocation: identifies mapping fovs 'close to' the target fov (using Look Up Tables (LUT)).

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

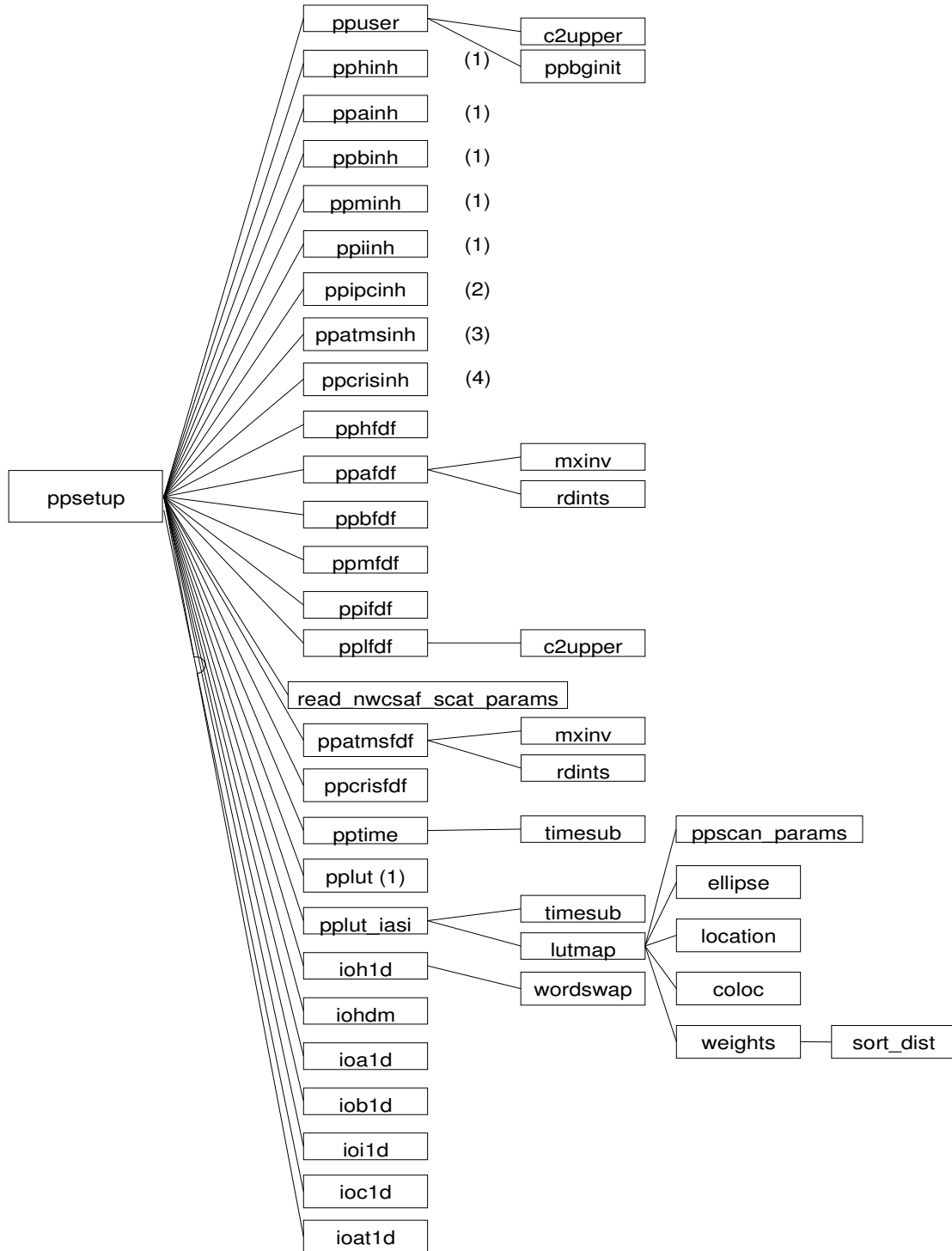
Estimation: calculates representative values (weights) of the mapping data at the target fov, using results of the colocation.

The fields of view of the two instruments create a pattern that repeats at regular intervals. This pattern is derived and then stored within a LUT which provides the location information.

After processing data become level 1d data.

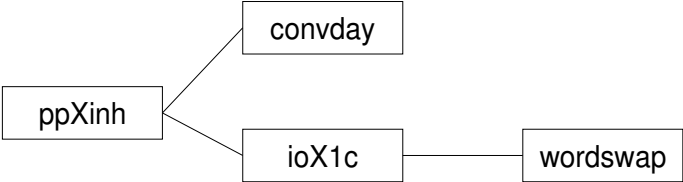
If you need to generate output products at different ATMS resolutions, be sure to take a copy of the original level 1c file.

TASK 1: INITIALISATION



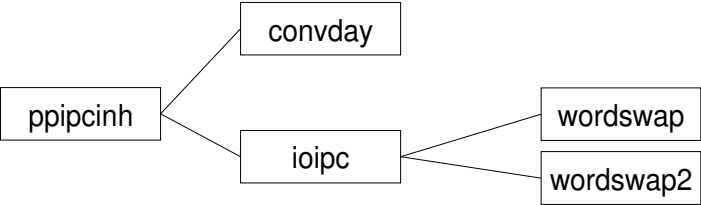
(1) (2) (3) (4) See its own modules hierarchy

(1)

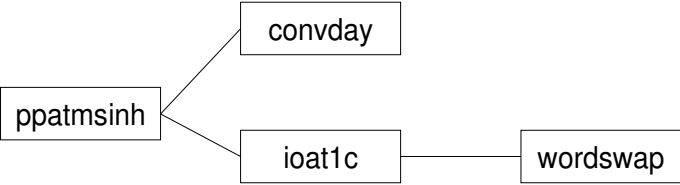


with X= h, a, b, m or i

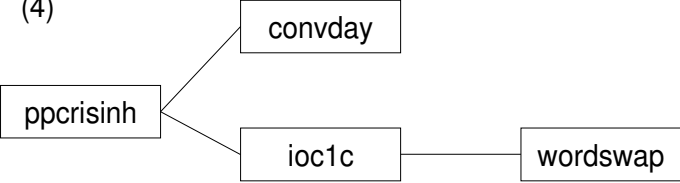
(2)



(3)

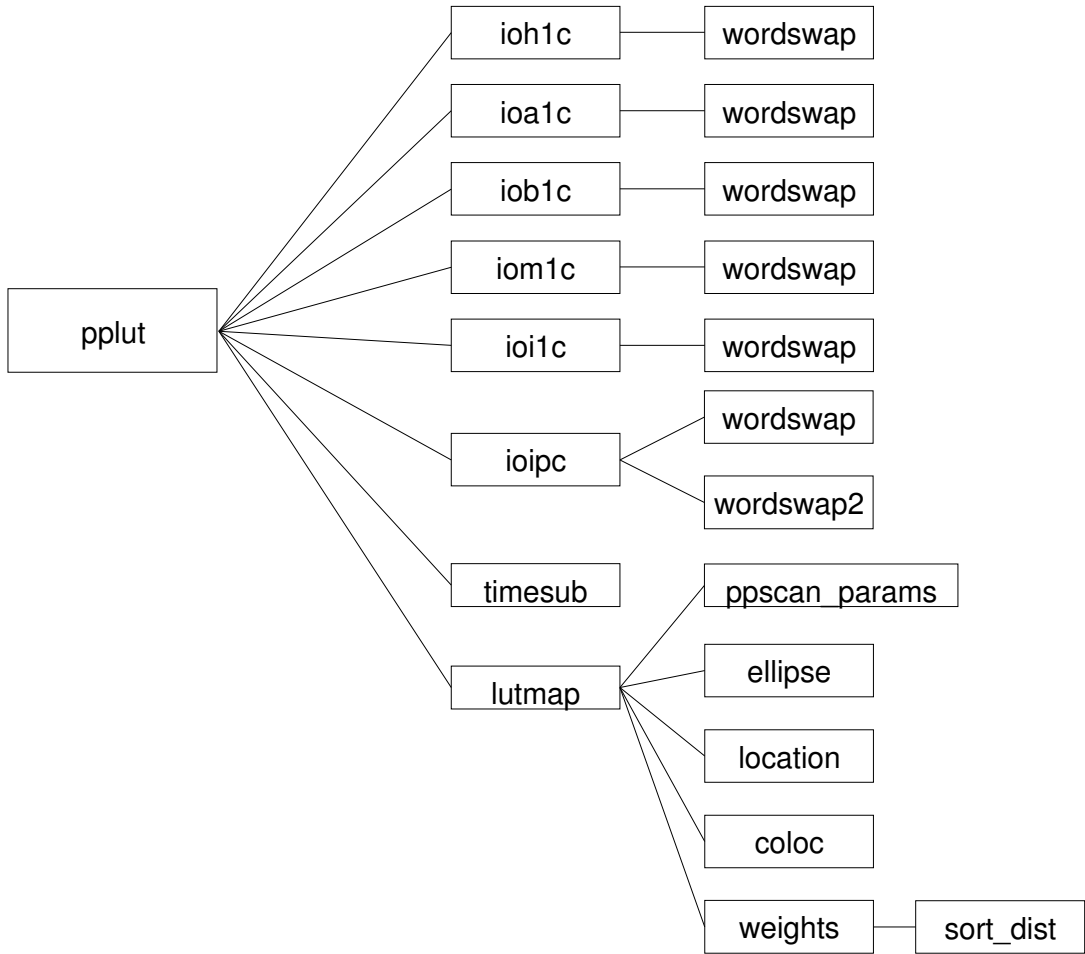


(4)



**Figure 4-19 : PPSETUP modules hierarchy**

To simplify the diagram, calls to the **errorreport** subroutine have not be written



**Figure 4-20 : PPLUT modules hierarchy**

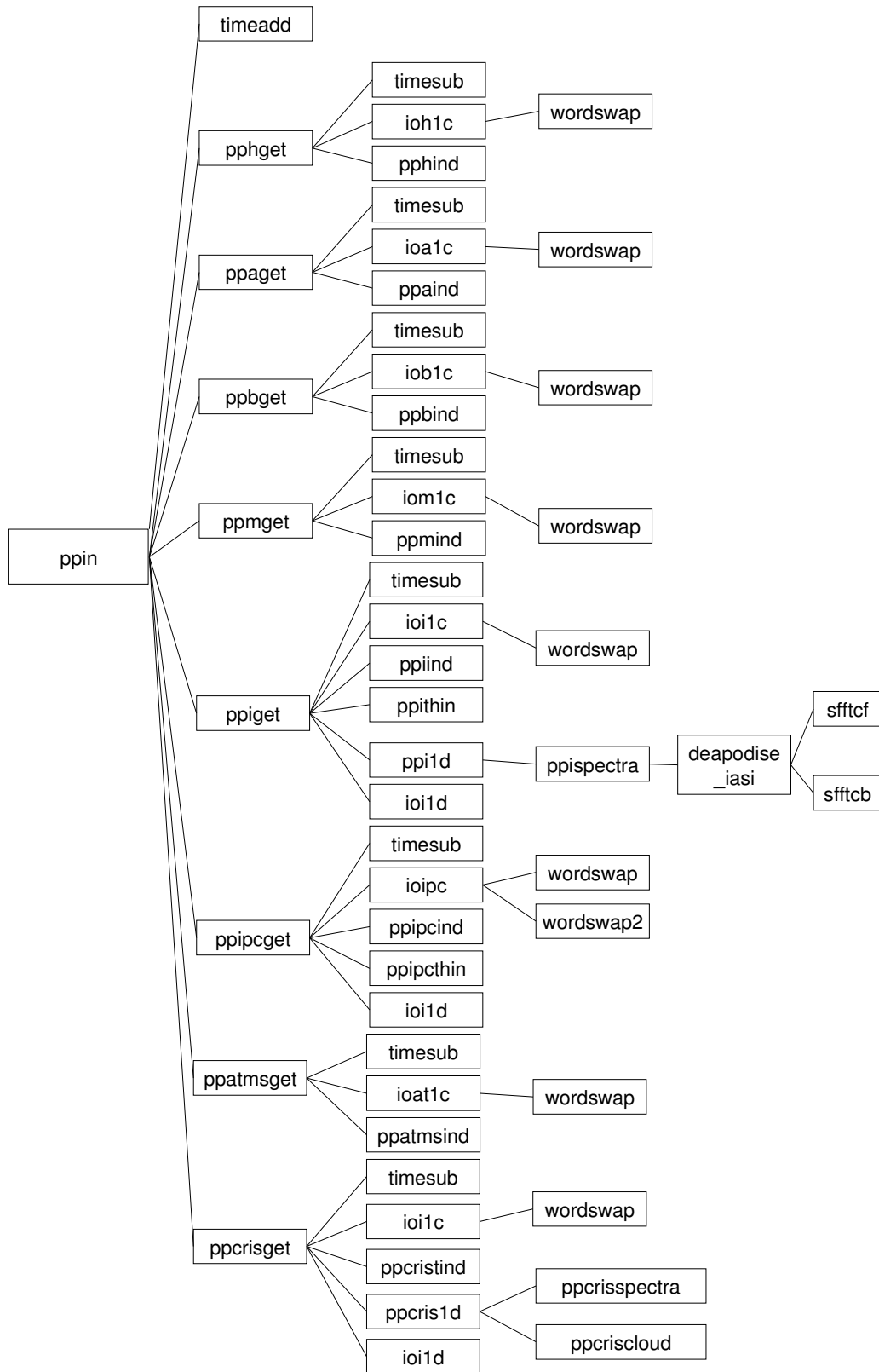


Figure 4-21 : PPIN modules hierarchy

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

This task performs all set up operations required for program **ATOVPP** (**ppsetup**).

First it reads user inputs (choice of mappings etc.) (**ppuser**). **ppuser** also defines unit numbers for all I/O, and calls **ppbginit** if Backus-Gilbert convolution has been selected for the AMSU-B to AMSU-A mapping.

Then it reads the level 1c headers and stores them in memory according to user inputs. One header-reading subroutine **ppXinh** corresponds to one instrument (X=a for AMSU-A, b for AMSU-B, m for MSU, h for HIRS, i for IASI, atms for ATMS and cris for CrIS).

It reads fixed data files and sets up fixed variables for each sounder (**ppXfdf**). Those data are described in the next chapter (it can be corrections to apply, parameters useful to processing, etc). There is a particular fixed data file for mapping ('LUT.fdf'), containing optional corrections and adjustments to perform for LUT initialisation. This file is read by the subroutine **pplfdf**, and its data are used by the subroutine **pplut**.

In the case of IASI, in addition to reading the IASI.fdf file, **ppifdf** also reads the files of reference eigenvectors (for Principal Components analysis). These are normally generated by EUMETSAT and distributed in HDF5 format. They include the noise normalization function. If required, a file giving the data required to transform from gaussian apodisation to self-apodisation can be read (referred to as an MTF – modulation transfer function – correction).

**ppsetup** also reads the AMSU-B scattering parameters that are used for the NWCSAF scattering and precipitation indices (**read\_nwcsaf\_scatter\_params**).

It then calculates start/end date/times for the processing and computes the number of blocks of data to process (**ppptime**). Data are processed in blocks of time interval "dt". "dt" should be flexible, but the intention is that it should be as long as possible within memory limitations. It could be one complete overpass for locally received data (~15 minutes) or even one complete orbit for global data (~100 minutes), but for small machines it may be less than these. The value of "dt" is set in the include file 'ppparms.h'.

Before pre-processing there is the creation of a LUT for each instrument by calling **pplut**. The main task of this subroutine is performed by the subroutine **lutmap**. It identifies those mapping fovs which are 'close to' (colocated with) target fovs (calls internal subroutines **ellipse**, **location** and **coloc**). The LUT also provide a representative value (a weight) for each mapping fov (internal subroutine **weights**) and for the appropriate mapping mode (nearest neighbour, bilinear interpolation (weighted average with the 4 corners), or spatial average (gaussian function or linear)). The resulting weight is applied to each colocated mapping fov to provide the mapped value. An appropriate LUT must be produced before running mapping routines. (Weights for Backus-Gilbert convolution are pre-calculated, and are read earlier by **ppbginit**. In the case of IASI, if only a single detector of the four is to be used then a call to **pplut\_iasi** is made at this point. If all detectors are to be used then the call to **pplut\_iasi** is delayed until later (see Task 3).

The mapping from ATMS to CrIS is performed using the actual geolocation latitudes/longitudes rather than look-up tables – see [37].

The following tasks (2, 3, 4 and 5) are performed on data blocks extracted from each instrument and stored in specific common blocks. This is done by calling subroutine **ppin** which reads data from each instrument (**ppXget**) according to input options and stores them in instrument-specific commons (**ppXind**).

For IASI and CrIS, **ppiget/ppcrisget** do not attempt to store in memory all the channel data for a block. Instead they read in the data for a scan line, performs IASI/CrIS-specific pre-processing (see below), then write the data for that scan line to the output level 1d file. The pre-processing steps are as follows:

- If spatial thinning has been requested, determine the “best” detector to use for each spot (**ppithin/ppcrisithin**).

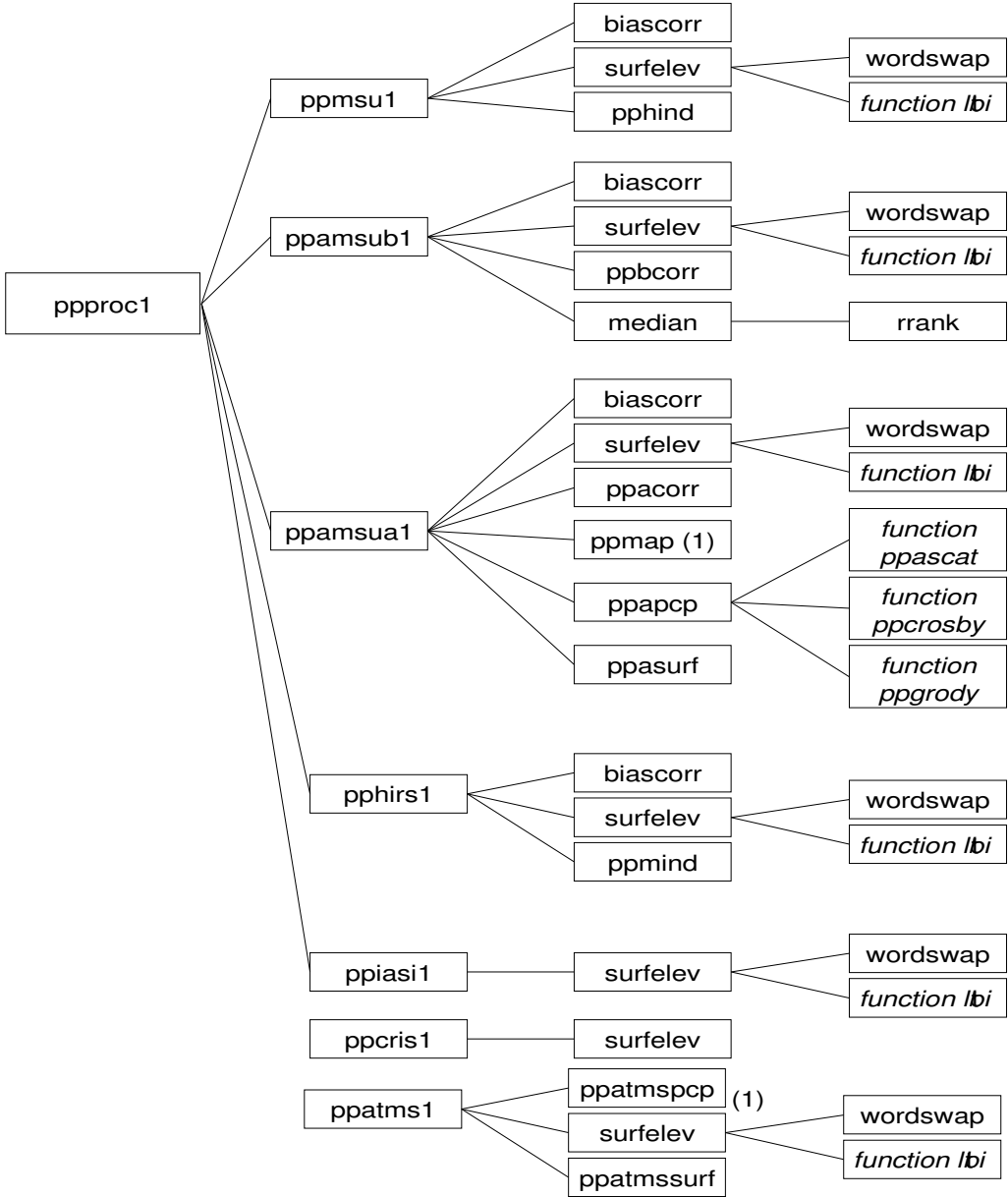
<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

- Copy selected data to the 1d common area (**ppi1d/ppcris1d**), e.g. latitude, longitude, radiance data for the channel selection specified in *IASI.fdf/CRIS.fdf* and, in the case of IASI, mapped AVHRR data.
- Compute Principal Component scores if this has been requested by the user (**ppispectra**, called from **ppi1d**; **ppcrisspectra**, called from **ppcris1d**).
- Write data to 1d file (**io1d/ioc1d**)

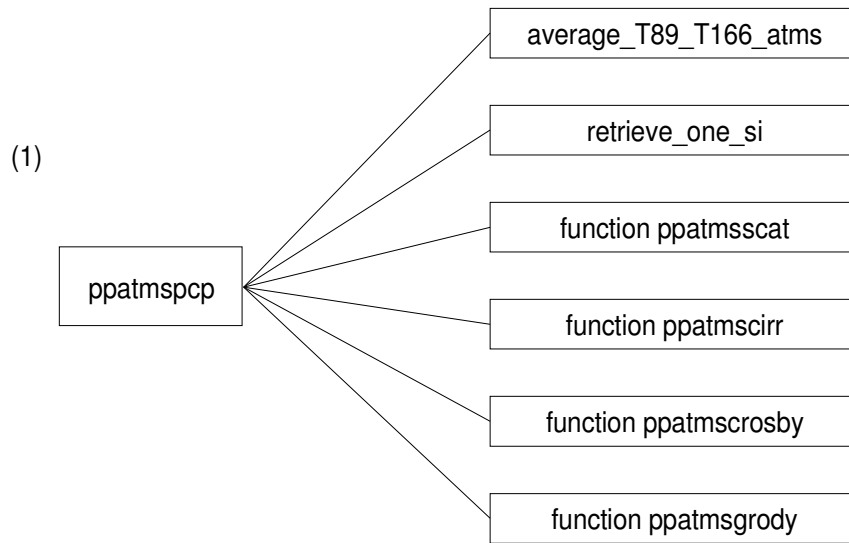
For the data block, the only IASI/CrIS data retained in memory (**ppiind/ppcrisind**) are the data that are required in the AMSU mapping process, i.e. the scan line numbers, scan line times, latitudes, longitudes and zenith angles.



**TASK 2: PRE-PROCESSING BEFORE MAPPING (PPPROC1)**



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---



**Figure 4-22 : PPROC1 modules hierarchy.**

To simplify the diagram, calls to the **errorreport** subroutine have not been written.

This task pre-processes a block of level 1c ATOVS data to level 1d before the mapping. The pre-processing differs for each ATOVS sounder, but has a common part. So, the main subroutine **ppproc1** calls a specific routine for each sounder instrument (**ppXXXX1** where XXXX = msu, amsua, amsub, hirs or iasi)

The common part of the pre-processing (general pre-processing) consists in performing bias corrections by calling subroutine **biacorr**. It adds a scan-dependent bias correction to level 1c brightness temperature. Values are added only where the BTs are not set to missing (i.e. within 4-400 Kelvin). Those values are read in the instrument-specific fixed data file, they are channel and scan-position dependent. If values are not provided in the instrument 'fixed data file' then values of zero are used.

Then, general pre-processing consists of extracting surface type and elevation for each fov by calling subroutine **surfelev**. It returns surface type (land/sea/mixed) and elevation for an array of lat/lon points, using the ITPP export package topography datasets (1/6th degree x 1/6th degree lat/lon, heights in 100s of feet). Height is for the nearest grid point, and is set to zero for sea spots. Surface type is found by examining all points within a box centred on the instrument field of view and approximately the same size as the field-of-view. The fov is classed land (or sea) only if every point in this box is land (or sea). Otherwise, the surface type is 'mixed'. If either latitude or longitude are out of range then values of -999999 are returned for both surface type and surface elevation.

For HIRS and IASI data no further pre-processing is done (**pphirs1**, **ppiasi1**). Subroutine **pphcorr** called at the end of the HIRS processing is a dummy routine and actually does nothing. It will correct 'limb' effects and surface emissivity.

For MSU data the next pre-processing step (**ppmsu1**) consists of subtracting the limb darkening curve read in the instrument fixed data file. This curve represents the expected difference (Kelvin) between the MSU brightness temperature of each HIRS fov and the one at nadir. There are two curves, one for the land and one for the sea. The aim is to help MSU mapping to HIRS (see scientific documentation). These curves are defined for each channel and for each HIRS scan angle. They are subtracted from MSU brightness temperatures at the MSU fofs before mapping,

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

and added back at the HIRS fofs after mapping. This may reduce errors in the mapping. The subroutine **ppmcorr** called at the end of the MSU pre-processing is a dummy routine doing nothing presently. It will correct 'limb' effects and surface emissivity.

For AMSU-B data during the pre-processing step (**ppamsub1**), a median filter (**median**) is applied to the 89GHz channel to detect spikes which may reveal contaminated data (e.g. due to scattering). The central fov of the 3x3 box is flagged if it differs by more than 10K. Note that we do not act on this flag in mapping AMSU-A to AMSU-B. The subroutine **ppbcorr** called during the AMSU-B pre-processing is a dummy routine that does nothing. It will correct 'limb' effects and surface emissivity.

A more important pre-processing is applied to AMSU-A data (**ppamsua1**). Note that some pre-processing on the AMSU-A grid uses mapped brightness temperatures from AMSU-B. This mapping (**ppmap**) is done within the AMSU-A pre-processing, but is described below in the next task.

Some precipitation tests are performed during the AMSU-A and ATMS data pre-processing by calling the subroutines **ppapcp** and **ppatmspcp**. They look for precipitation signals in AMSU-A/ATMS and set flags accordingly.

The following paragraphs describe AMSU-A processing; there are equivalent routines for ATMS. First, a scattering test (**ppascat**) is performed by computing and checking the scattering index. This test can only be used over the sea. It consists in estimating the AMSU-A channel 15 brightness temperature (BT) from channels 1, 2 and 3, and then determining the scattering index by differencing the observed and computed BT15. If the scattering index is > 10k or <-10K the fofs are flagged as scattering (see scientific documentation).

Then the Crosby logistic precipitation test [6] is performed by calling subroutine **ppcrosby**. This test is applied to AMSU-A channels 1 & 15 returns the probability of rain. This test, which is also only applicable over the sea, also uses the relative scattering by hydrometeors at high frequency to flag rain or deep ice cloud. The information is very similar to the scattering index and so this test may be redundant.

Lastly, the Grody light rain test is performed by calling **ppgrody**. This test is applied to the AMSU-A channels 1 & 2. It returns 'TRUE' if rain is detected (see scientific documentation).

After the precipitation tests, the remaining AMSU-A pre-processing consists of estimating the surface type of each fov from the brightness temperatures (**ppasurf**) using only selected channels 1, 2 and 3. The following surface types can be detected :

- 1 = Bare young ice (i.e. new ice, no snow)
- 2 = Dry land (i.e. dry with or without significant vegetation)
- 3 = Dry snow (i.e. snow with water content less than 2%, over land)
- 4 = Multi-year ice (i.e. old ice with snow [assumed dry] cover)
- 5 = Sea (i.e. open water, no islands, ice-free, wind < 14m/s)
- 6 = Wet forest (i.e. established forest with wet canopy)
- 7 = Wet land (i.e. non-forested land with a wet surface)
- 8 = Wet snow (i.e. snow with water content > 2%)
- 9 = Desert.

**Note** : If surface type is 1, 4 or 8 and channel 1 > 275K surface type is set to 9.

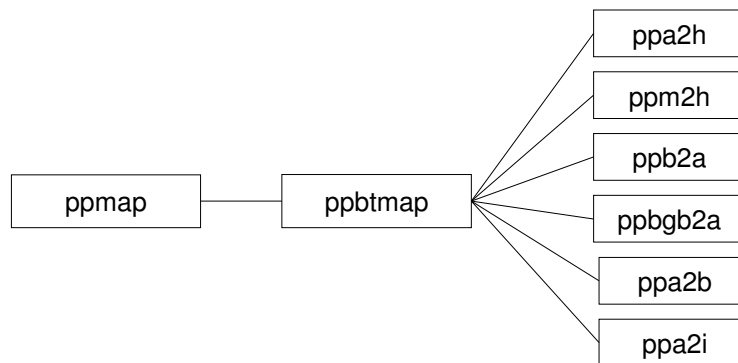
Flags are set if:

- the minimum value of the cost function exceeds the cloud-test threshold
- the estimated surface type is incompatible with topography
- the surface type is one which cannot be processed in the next steps (2, 3, 6, 7 and 9)

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

The subroutine **ppacorr** called during the AMSU-A pre-processing is a dummy routine that does nothing. It will correct 'limb' effects and surface emissivity.

TASK 3: MAPPING INSTRUMENTS (PPMAP)



**Figure 4-23 : PPMAP modules hierarchy.**

This task maps data from one instrument grid to another via the subroutine **ppmap** which calls **ppbtmap** for each mapping to process.

For most instruments **ppmap** is called once per data block. However, for IASI **ppmap** and **pplut\_iasi** are called up to four times per block – once for each IASI detector.

**ppbtmap** maps brightness temperatures between ATOVS instrument grids (Companion routine: **lutmap** see above). **lutmap** generates a look-up (LUT) which identifies those mapping fovs which are colocated with a target fov. The LUT also provides a weight for each mapping fov for each mapping mode (if bilinear interpolation or spatial average). The weights for the selected mode are applied to the corresponding BTs and the resulting sum provides the mapped value.

**Note** that a set of several observations is mapped with one call to **ppbtmap**.

Five mappings are available :

1. AMSU-A to HIRS (**ppa2h**).
2. MSU to HIRS (**ppm2h**).
3. AMSU-B to AMSU-A (**ppb2a** or **ppbgb2a**).
4. AMSU-A to AMSU-B (**ppa2b**).
5. AMSU-A to IASI (**ppa2i**).

The subroutines **ppa2h** and **ppa2i** map AMSU-A fovs to an individual HIRS or IASI fov. The routine selects mapping fovs from those given and derives brightness temperatures and other parameters at the specified target fov. After initialisation, the routine identifies 'good' mapping fovs by selecting only mapped fovs with valid brightness temperatures. It then calculates mapped BTs using weights from the LUT and finally tests if all AMSU-A fovs have the same surface type and sets a flag accordingly.

**Note** : The method used here was considerably simplified from the one used in the earliest versions of AAPP, in which care was taken only to map AMSU-A fovs with the same surface type and cloud classification. The original method would often find only one suitable AMSU-A fov for each of several HIRS fovs, and so re-use it several times. The resulting mapped values were then "blotchy". The current method chooses all nearby AMSU-A fovs with a valid BT. It

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

takes no account of surface type and cloud in the mapping, but flags are set if the AMSU-A fovs are not of identical surface type and if any is flagged for cloud.

The subroutine **ppm2h** maps MSU fovs to an individual HIRS fov. It selects mapping fovs from those given and derives brightness temperatures and other parameters at the specific target fov. After initialisation, the routine first tries only those fovs with primary calibration. Otherwise it accepts those with secondary calibration. It sets flags and surface types before the calculation of mapped BTs. Note that MSU fovs are accepted irrespective of surface type, however a flag is set if the surface type of any of those selected differs from that of the HIRS fov.

The subroutine **ppb2a** (or **ppbgb2a**) maps AMSU-B fovs to an individual AMSU-A fov. It selects mapping fovs from those given and derives brightness temperatures and other parameters at the specific target fov. After initialisation, the routine first tries only those fovs with primary calibration. Otherwise it accepts those with secondary calibration. It checks the range of the 89GHz channel over the AMSU-A fov and finishes setting the flags and calculates mapped BTs. Note that only those AMSU-B fovs with the same surface type as the AMSU-A fov are mapped, unless AMSU-A fov is of mixed type, in which case all AMSU-B fovs are mapped.

The subroutine **ppa2b** maps values from AMSU-A grid to AMSU-B fov. Note that we are using the nearest AMSU-A fov only so this is a simple task. After initialisation, the routine derives brightness temperatures from AMSU-A to AMSU-B, and then maps pre-processing variables.

The mapping of ATMS to CrIS is performed by a separate subroutine **map\_atms\_to\_cris**, called directly from the atovpp main program. It does not use **ppmap**.

#### TASK 4: PRE-PROCESSING AFTER MAPPING (PPPROC2)

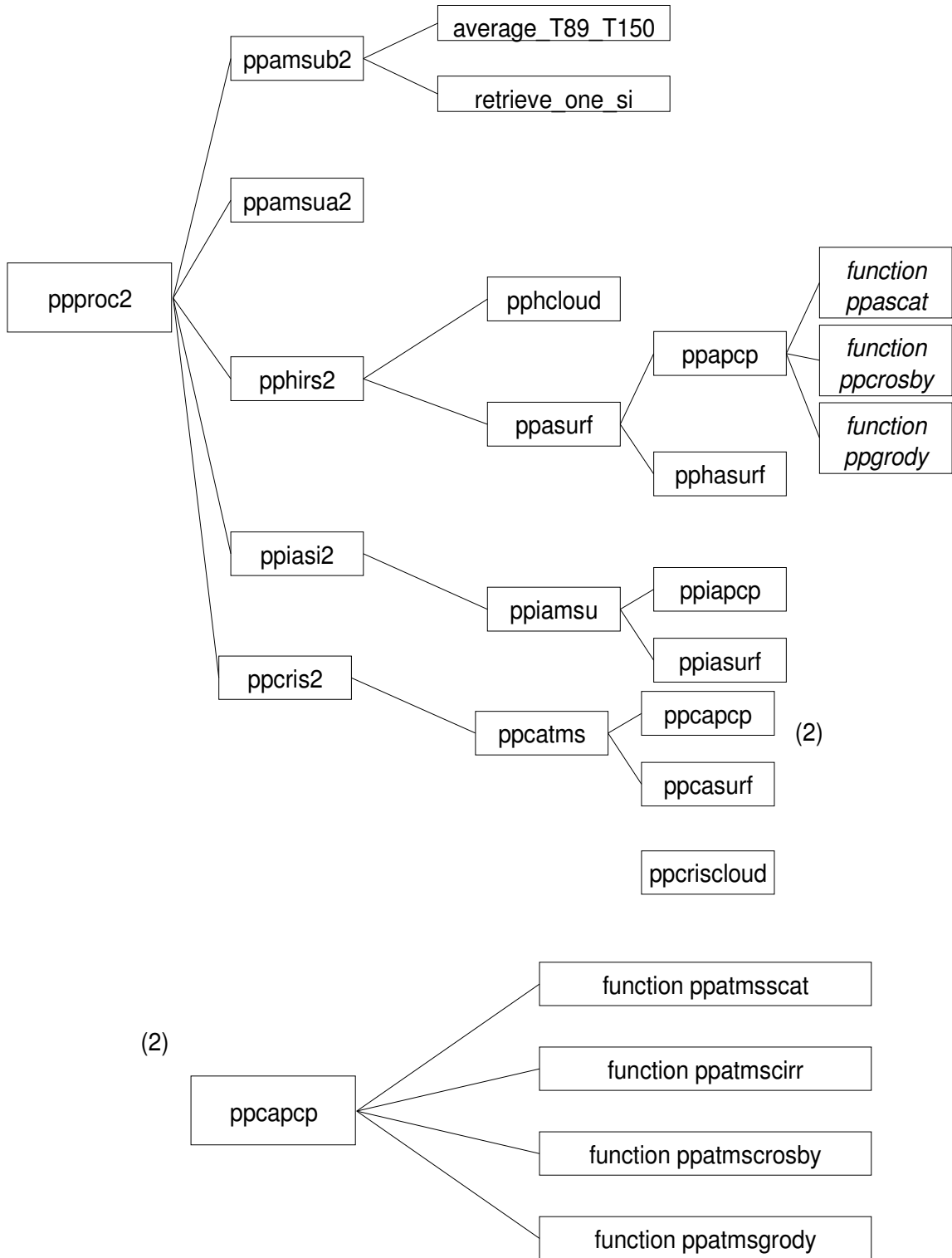


Figure 4-24 : PPPROC2 modules hierarchy.

To simplify the diagram, calls to the **errorreport** subroutine have not been written

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

This task pre-processes a block of level 1c ATOVS data to level 1d after mapping for each instrument.

The following pre-processing options are available :

1. AMSU-B pre-processing (**ppamsub2**)
2. AMSU-A pre-processing (**ppamsua2**)
3. HIRS pre-processing (**pphirs2**)
4. IASI pre-processing (**ppiasi2**)
5. CrIS pre-processing (**ppcris2**)

Note that an instrument is only processed here if output has been requested on that instrument grid. Currently pre-processing option 2 does nothing.

The subroutine **ppamsub2** pre-processes a block of level 1c AMSU-B data after mapping. It presumes that AMSU-A brightness temperatures have been already mapped to AMSU-B. It recalculates (with **ppascat**) the AMSU\_A scattering index using the AMSU-B 89Ghz channel instead of AMSU\_A. It flags where the AMSU-A and 89GHZ channels differ. It then calls **ppacirr** to calculate the cirrus scattering index by estimating the AMSU-B 183GHz brightness temperature. Finally it computes the NWCSAF scattering and precipitation indices.

The subroutine **pphirs2** pre-processes a block of level 1c HIRS data after mapping. It first adds back the MSU limb darkening curves (different for land and sea) to HIRS fovs. These curves are defined for each channel and for each HIRS scan angle (see task 2 of 3.2.11). The subroutine then tests for cloud by calling the subroutine **pphcloud** (which currently does nothing). Lastly, **pphirs2** repeats pre-processing tests for AMSU-A but on the HIRS grid (**pphamsu**: see task 2 **ppamsua1**) :

- Looks for precipitation signal in AMSU-A mapped to HIRS grid and sets flags accordingly (**pphapcp**) : performs scattering test (**ppascat**), Crosby and Ferraro & Wu test (**ppcrosby**) and Grody light rainfall test (**ppgrody**).
- Estimates surface type and flags cloud liquid water using AMSU-A data mapped to the HIRS grid (uses AMSU-A and AMSU-B channels) by calling **pphasurf** (derived from **ppasurf** : see task 2).

The subroutine **ppiasi2** also repeats pre-processing tests for AMSU-A, but on the IASI grid (**ppiamsu**). Similarly, **ppcris2** repeats pre-processing tests for ATMS, but on the CrIS grid.

#### TASK 5: DATA WRITING TO 1D LEVEL FILES (PPOUT)

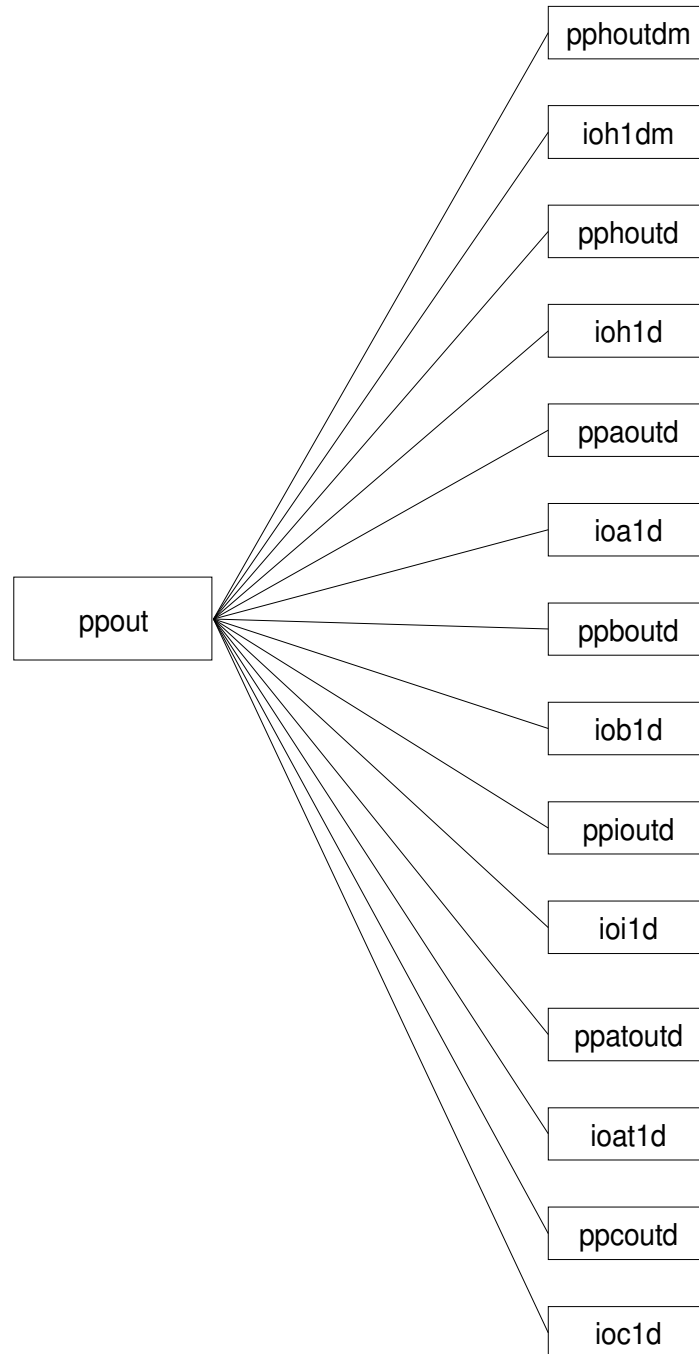


Figure 4-25 : PPOUT modules hierarchy.

This task creates level 1d records from memory stored values, and writes out to level 1d files (**ppout**). This program calls one different subroutine for each instrument to write :

1. HIRS (TOVS or ATOVS) (**pphoutdm** or **pphoutd**)
2. AMSU-A (**ppaoutd**)
3. AMSU-B (**ppboutd**)
4. IASI (**ppioutd**)
5. ATMS (**ppatoutd**)

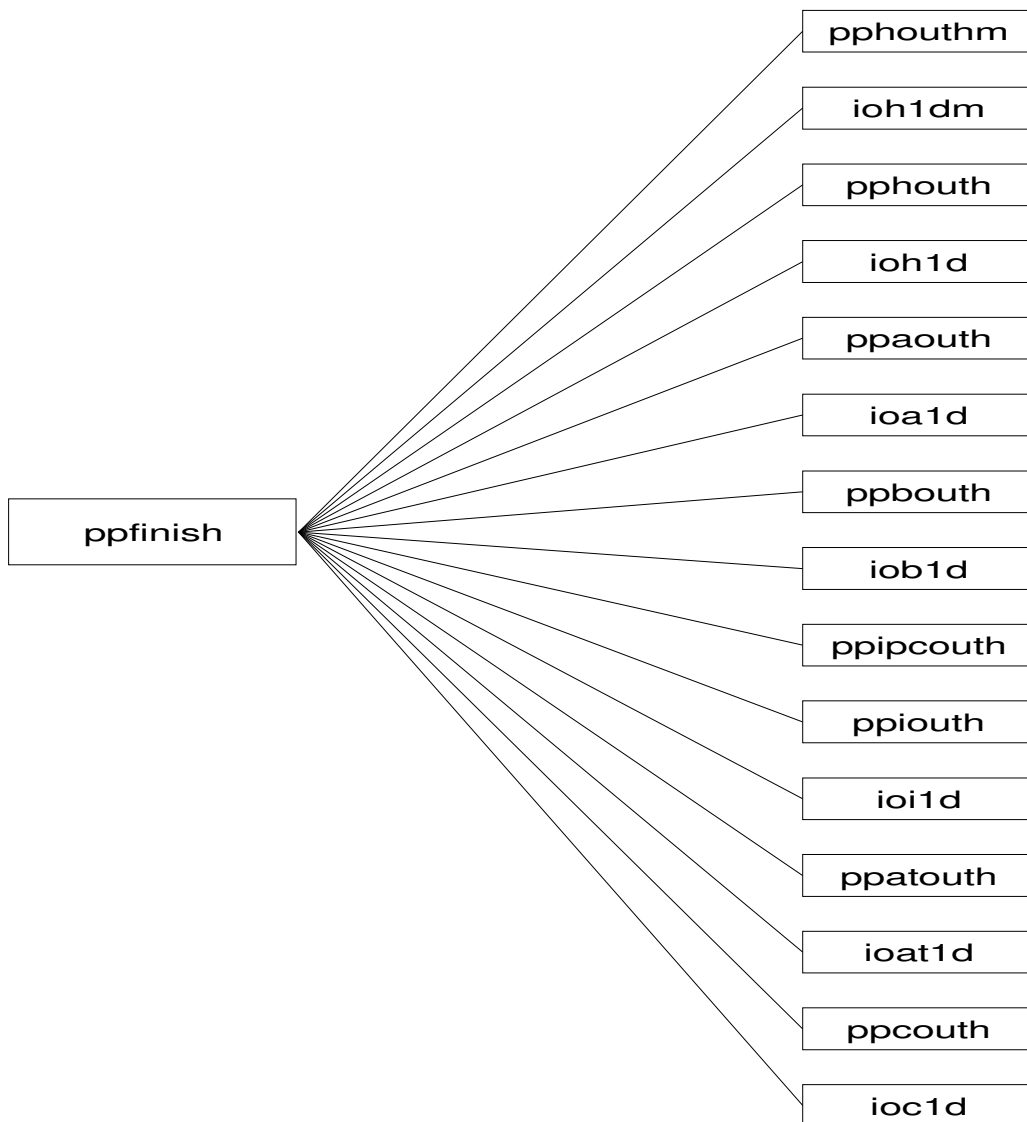


### 6. CrIS (**ppcouth**)

The subroutine **ppout** may overwrite the last record from the previous block, if the same scan line has been processed within this block. This is because the last scan line in a block is at a disadvantage in the pre-processing, e.g. when applying a horizontal filter. It is preferable to overwrite it with the same scan line from the next block. Similarly, the first scan line from the current block may not be written, if it was already processed as an "interior" line from the previous block.

**ppout** calls the subroutines **pphouthm**, **pphouth**, **ppaouth**, **ppbouth**, **ppiouth**, **ppatouth** and **ppcouth** to transfer data from program arrays to a level 1d data record, and then write out the record by calling I/O routine for level 1D data **ioX1d(m)** (where X = a for AMSU-A, b for AMSU-B and h for HIRS and with m added for TOVS data).

#### TASK 6: HEADER WRITING TO LEVEL 1D FILES (**PPFINISH**)



**Figure 4-26 : PPFINISH modules hierarchy**

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

This task writes out level 1d headers for each instrument. The main **ppfinish** calls one different subroutine for each instrument :

1. HIRS (TOVS or ATOVS) (**pphouth** or **pphoutm**)
2. AMSU-A (**ppaouth**)
3. AMSU-B (**ppbouth**)
4. IASI (**ppiouth**)
5. ATMS (**ppatouth**)
6. CrIS (**ppcouth**)

The subroutines **pphouthm**, **pphouth**, **ppaouth**, **ppbouth**, **ppiouth**, **ppatouth** and **ppcouth** have the same structure: set up the level 1d header using information from level 1c headers. Check that the format version number and data type level 1c header has already been read into common /xxx1chd/. Check that format of level 1c & level 1d include files (*xxx1c.h*, *xxx1d.h*) are compatible with the code of this subroutine (xxx=hrs for HIRS, ama for AMSU-A, amb for AMSU-B and iasi for IASI). Check that the format of level 1c file is compatible with this subroutine. Calls **ErrorReport** to print a warning if there is a problem. Lastly it sets up level 1d header. Writing is performed by calling the I/O routine for level 1d data **ioX1d(m)** (where X = a for AMSU-A, b for AMSU-B, h for HIRS and i for IASI, and m added if we process TOVS data).

#### **4.1.19. Modify the ATMS beam width: ATMS\_BEAMWIDTH script, ATMS\_BEAMWIDTH.EXE**

Modify the ATMS beam width for a level 1c file.

The input and output beam widths for each ATMS channel are specified in a data file given by environment variable \$ATMS\_BEAMWIDTH\_FILE (default atms\_beamwidth.dat, in \$DIR\_PREPROC). For more information on ATMS beam manipulation, see [37] document NWPSAF-MO-UD-027 (appendix to AAPP scientific documentation).

Note: If outfile is not specified then the input file is over-written..

#### **4.1.20. Modify the MWTS2 or MWHS2 beam width: MWTS2\_BEAMWIDTH and MWHS2\_BEAMWIDTH scripts, MWTS2\_BEAMWIDTH.EXE and MWHS2\_BEAMWIDTH.EXE**

Modify the MWTS2/MWHS2 beam width for a level 1c file.

The input and output beam widths for each channel are specified in a data file given by environment variable \$MWTS2\_BEAMWIDTH\_FILE (default mwts2\_beamwidth.dat, in \$DIR\_PREPROC), and similarly for MWHS2.

The method is similar to that used for ATMS. The default files specify 3x3 averaging.

#### **4.1.21. Mapping AVHRR to HIRS + Cloud Mask: AVH2HIRS script , AVH2HIRS.EXE or AVH2HIRS\_ATOVS.EXE.**

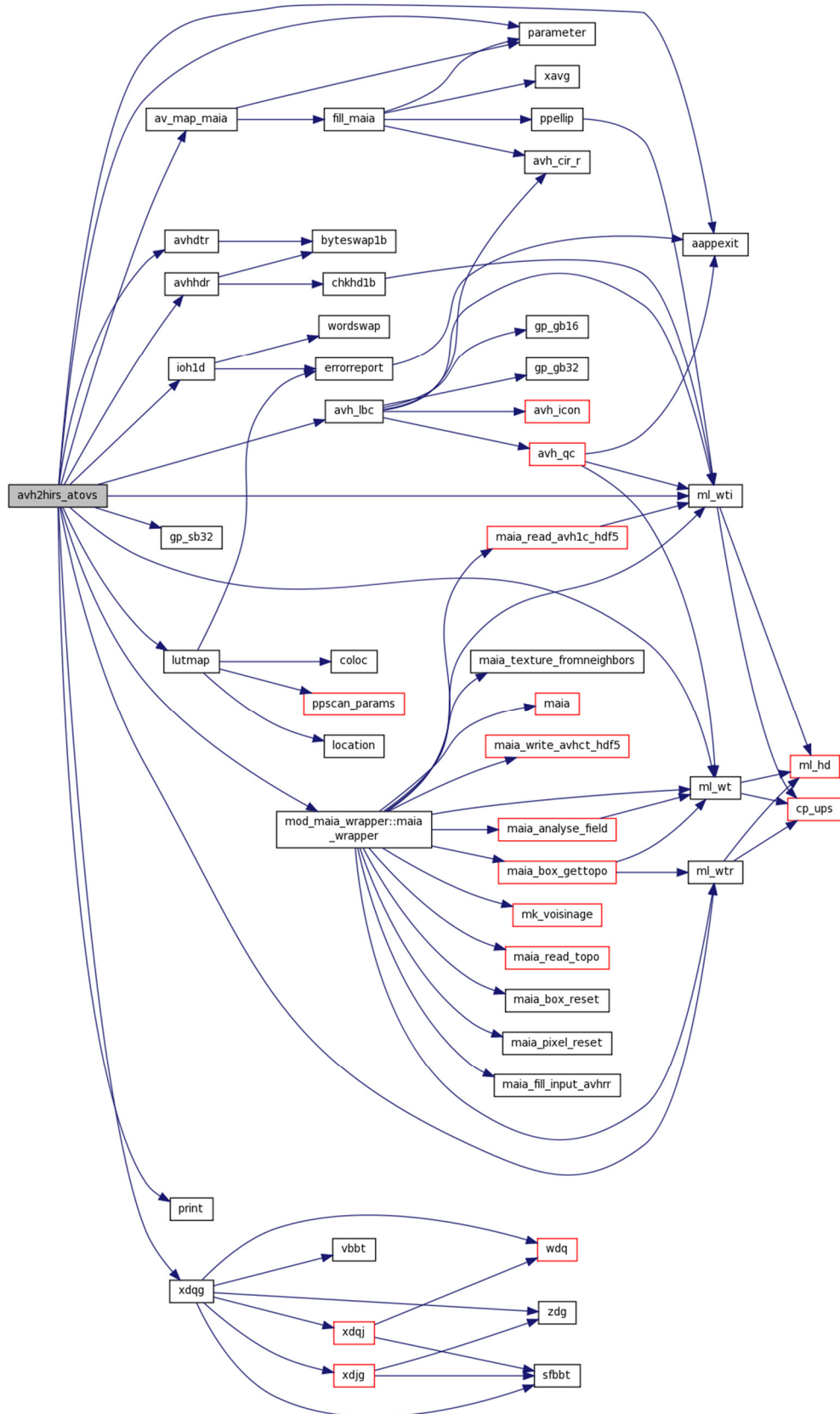


Figure 4-27 : AVH2HIRS\_ATOVS/AVH2HIRS modules hierarchy

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

This task requires the HIRS level 1d file , the AVHRR level 1b file, and several resource files.

### TASK 1: INITIALISATION

A part of the initialisation is directly coded inside the main program **AVH2HIRS** (**AVH2HIRS\_ATOVS**), particularly for parameters used in the LUT generation, information and options for mapping (e.g. mapping mode is set to 2 and local is set to true). The surface option for mapping is set to 0 (mapping with no surface type requirement).

A number of the thresholds used to determine the cloud mask are set up in constants included in the file *maia.h*.

Dynamic initialisations involve reading the input files (data, calibration coefficients, climatology, corrections etc.), storing information into arrays or commons, computing various parameters, and setting up the LUT for mapping. All of this information will be essential to the processing.

First, the AVHRR header buffer is set up by calling the subroutine **avhhdr** which reads AVHRR level 1b file. Then, for each HIRS fov, the HIRS level 1d header and data records are read (**ioh1dm** or **ioh1d**), and the extracted viewing geometry and surface information are stored in arrays ('*targ\_angles*' : latitude, longitude, solar and local zenith and azimuth angles ; '*targ\_alt*' : surface elevation ; '*targ\_surf*' : surface type).

The time and angle correction file for the LUT is then read and the extracted information is also stored in arrays ('*scan\_angle\_cor*' and '*time\_cor*').

Then the tasks required to initialise the '*tconv*' look-up table to convert radiance into brightness temperature for the 3 AVHRR infrared channels by calling the subroutine **avh\_icon** .

The initialisation of the cloud parameters (*maia*) is done by the optional call to the *maia\_wrapper* routine.

1.

If the HIRS level 1d file contain mapped AMSU-A data, over sea, **avh2hirs** (**avh2hirs\_atovs**) computes the total water vapor content with channels 23, 31 and 50Ghz and fills the common */wv\_sat/*.

Lastly the initialisation of the look-up table (buffer '*lutbuf*') for mapping AVHRR to a HIRS data block (5 lines) is performed by calling the subroutine **lutmap**. Then it computes the minimum and maximum AVHRR line numbers for the 5 HIRS lines (corresponding to a block of HIRS data).

**Note** : the following tasks(2, 3) process an HIRS block of data.

### TASK 3: MAPPING

This task is performed by the subroutine **av\_map\_maia**.

First for each target line, it calculates the line number relative to the look-up table from *lutmap*, and then calculates a mapping line offset between the mapping LUT line and the level 1b mapping line.

For all target fovs in the line, it creates three box arrays (pixels x lines) for AVHRR brightness temperatures ('*box\_bts*'), radiance boxes ('*box\_rads*') and mapping ('*box\_map*'). These boxes are centred on the HIRS target pixel ( 33 x 38 is a good size to include HIRS fov). Values of the mapping box are as follows : 0 = pixel inside the ellipse, 1 = outside, 2 for AVHRR missing line or bad pixels.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

For each HIRS target pixel, the **maia** cloud mask subroutine is called with the three box arrays in input. The mask is based on a threshold technique applied to every AVHRR pixel inside the HIRS ellipse. Threshold tests are applied to various combinations of channels. If the succession of tests is successful the pixel is considered as 'clear'. The combinations of channels used depend on the geographical location of the pixel (land, sea, coast), on the solar illumination, and the viewing geometry (daytime, night-time, dawn, sunglint). The thresholds are computed with empirical functions (of viewing angle, pixel BTs, total water vapour content of the atmosphere...), from climatological datasets of SST, albedo and specific humidities, and from NWP outputs (surface air temperature over land and twvc).

The longitudes for climatologies and forecast are systematically converted in the range [-180, 180].

#### Maia algorithm is described in section

Once the mask is applied, statistics are computed for 'clear' pixels : averages (*'tmoy\_cl'*) for each channel (**xavg**) and the channel 4 standard deviation (*'std4\_cl'*). Using that, the 13 parameters of the local mode are computed for the HIRS target fov and stored into the array *'targ\_bts'*. These 13 parameters are as follow :

1. percentage clear AVHRR in HIRS FOV (\*100)
2. surface temperature (K\*100)
3. climatological temperature or t2m (K\*100)
4. mean AVHRR channel 3 over HIRS FOV (degK\*100)
5. mean AVHRR channel 4 over HIRS FOV (degK\*100)
6. mean AVHRR channel 5 over HIRSFOV (degK\*100)
7. black body coverage in HIRS FOV (degK\*100)
8. top cloud temperature over HIRS FOV (degK\*100)
9. std top cloud temperature over HIRS FOV (degK\*100)
10. mean clear AVHRR channel 4 over HIRS FOV (degK\*100)
11. mean clear AVHRR channel 5 over HIRS FOV (degK\*100)
12. std AVHRR channel 4 over HIRS FOV (degK\*100)
13. std clear AVHRR channel 4 over HIRS FOV (degK\*100)

#### TASK 5: WRITING OUTPUT FILES

Each HIRS data line is read from HIRS level 1d file and stored in the corresponding 1d common (**ioh1dm** or **ioh1d**). Then, for each of the 56 fofs of the HIRS target line, the 13 AVHRR parameters are set up with corresponding values of the array *'targ\_bts'*. The result is stored in a buffer (*hrs1d\_avhrr(56,13)*) included in the common of the HIRS 1d line. This common is then written into the HIRS level 1d record (**ioh1dm** or **ioh1d**).

For each target fov, statistics are computed on the difference between brightness temperatures (BTs) of the HIRS channel 8 (H8) and mean BTs of the AVHRR channel 4 (A4). First, the following calculations are made on H8-A4 :

- sum for each column and total sum
- sum squared for each column and total sum squared.
- count for each column and total count

Then the following calculations are made :

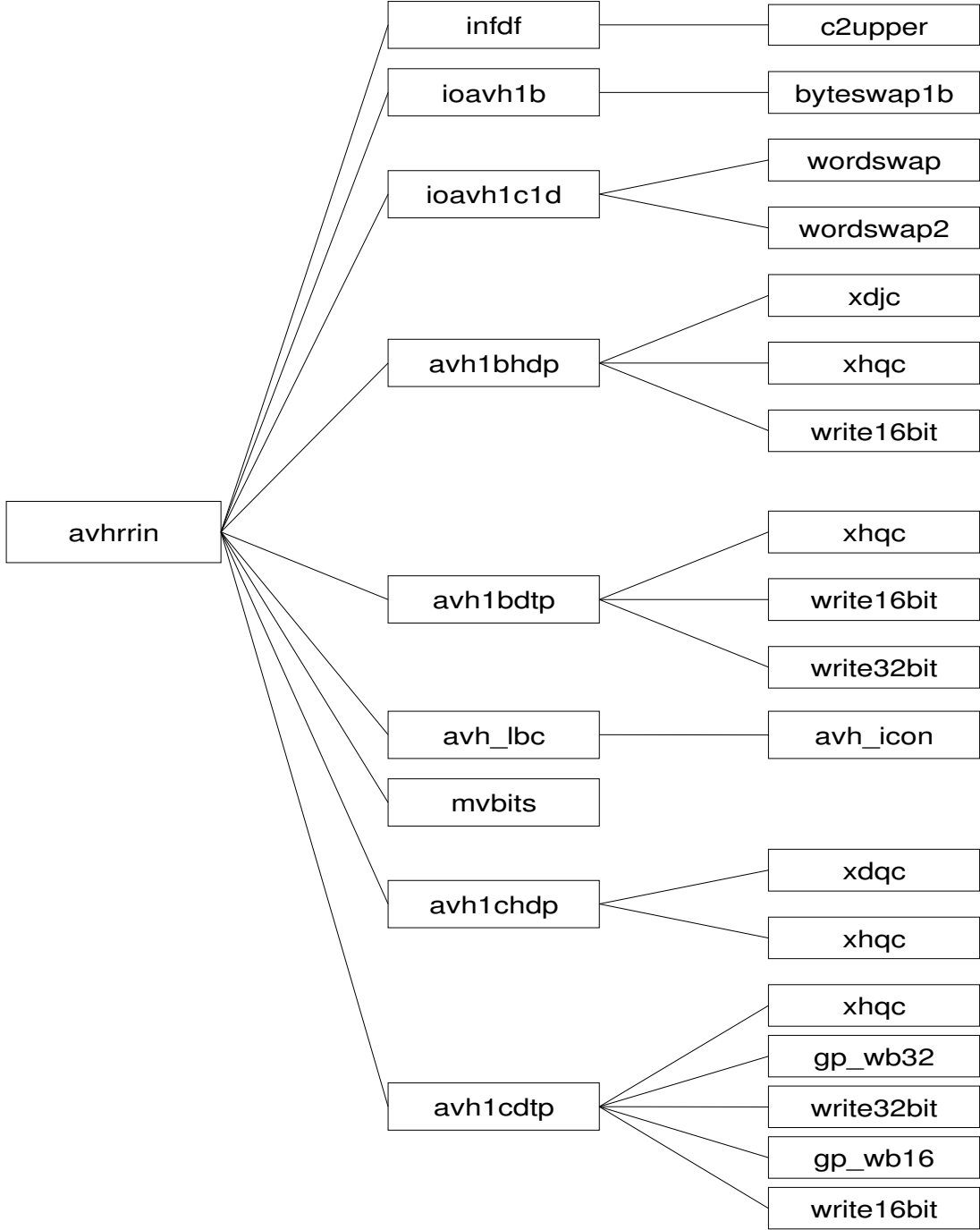
- average for each column and total average
- standard deviation for each column and total standard deviation

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

Then it writes to the standard output, and writes standard deviation for each column and total standard deviation to a formatted historical file.

Lastly, it writes AVHRR quality information to standard output ('good' and 'bad' lines, missing line etc.), and closes AVHRR level 1B and HIRS level 1D files.

**4.1.22. AVHRR calibration: AVHRRIN script and AVHRRIN.EXE**



**Figure 4-28 : AVHRRIN modules hierarchy**

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

This task requires the AVHRR level 1b file and the *fdf.dat* file. It applies calibration coefficients computed by **avhrc1** to output counts to produce reflectances and radiances. Then it performs radiance conversion to brightness temperature. The output file is an AVHRR level 1C file.

#### TASK 1: INITIALISATION

The program reads the input data and the options.

After it defines the bit numbers. The convention used in 1B & 1C files is that an INTEGER\*4 word has bits numbered 0-31, with bit 0 being the least significant bit. Some platforms take bit 31 as the LSB. It is necessary to define the order of bits that we use, to keep the code portable.

Various tests are used.

Then the program reads the fixed data file (**call infdf**).

#### Task 2: AVHRR CALIBRATION

This task begins by opening the input and output files. It reads the header of the input file (**ioavh1b**) and sets up the header of the output file for writing it (**ioavh1c1d**).

After, it goes through all scan lines, reading (**ioavh1b**), applying calibration coefficients (**avh\_lbc**), writing into the output file (**ioavh1c1d**)

To finish, the files are closed (**ioavh1b** and **ioavh1c1d**).



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### **4.1.23. Convert AVHRR AAPP I1b format to AVHRR PFS L1B format: AAPP-EPS AVHRRL1B script and EPS\_AVHRRL1B-MAIN.EXE.**

This script and its attached binary program converts AVHRR encoded in AAPP format to AVHRR in PFS 6.5 format.

Only a partial conversion is achieved, that is, only fields required by IASI OPS are filled:

- MPHR :
- PARENT\_PRODUCT\_NAME\_1
- INSTRUMENT\_ID
- INSTRUMENT\_MODEL
- PROCESSING\_LEVEL
- SPACECRAFT\_ID
- PROCESSING\_CENTRE
- RECEIVING\_GROUND\_STATION
- SENSING\_START
- RECEIVE\_TIME\_START
- SENSING\_START\_THEORETICAL
- SENSING\_END
- RECEIVE\_TIME\_END
- SENSING\_END\_THEORETICAL
- TOTAL\_MPHR
- TOTAL\_SPHR
- TOTAL\_GIADR
- TOTAL\_RECORDS
- DURATION\_OF\_PRODUCT
- MILLISECONDS\_OF\_DATA\_PRESENT
- PROCESSING\_TIME\_START
- PROCESSING\_TIME\_END
- PRODUCT\_NAME
- SPHR
- EARTH\_VIEWS\_PER\_SCANLINE
- NAV\_SAMPLE\_RATE
- MDR\_1B
- EARTH\_VIEWS\_PER\_SCANLINE
- NUM\_NAVIGATION\_POINTS
- DIGITAL\_B\_DATA
- FRAME\_INDICATOR
- CALIBRATION\_QUALITY
- SCAN\_LINE\_QUALITY
- NAVIGATION\_STATUS
- SCENE\_RADIANCES
- EARTH\_LOCATIONS
- EARTH\_LOCATION\_FIRST
- EARTH\_LOCATION\_LAST
- ANGULAR\_RELATIONS

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

- TIME\_ATTITUDE
- EULER\_ANGLE
- SPACECRAFT\_ALTITUDE
- COUNT\_ERROR\_FRAME

This program auto-detects the endianness of the AAPP input file. It make call to avh\_lbc to convert digital data to radiances. Geolocation data is interpolated from 51 to 103 points.

Data is read sequentially from AAPP format and rewritten to a PFS file.

#### **4.1.24. Convert IASI PFS L1C to IASI AAPP I1c : CONVERT\_IASI1C, CONVERT\_IASI1C.EXE and CONVERT\_IASI1C\_9.0.EXE**

This program converts a IASI 1C PFS file in a AAPP IASI 1C file.

Data are read sequentially from the PFS file and written to AAPP format using the following Fortran subroutines:

- open1c : open AAPP file
- mdr1c : converts and writes a IASI 1c record
- mphr : converts and writes a IASI 1c record
- giadr : extracts information from GIADR record
- finish1c : close AAPP file

#### **4.1.25. Convert NOAA I1b formats to AAPP I1b format: noaa\_class\_to\_aapp script and associated executables**

Introduced in AAPP v7.6. The script **noaa\_class\_to\_aapp** ingests level 1B files from the NOAA archives and outputs AAPP level 1B format. The following formats and instruments are supported:

Tiros-N to NOAA-14:

- MSU
- HIRS/2
- AVHRR and AVHRR/2 LAC and GAC 10-bit format

NOAA-15 to NOAA-19:

- AVHRR/3 LAC and GAC 10-bit and 16-bit formats
- The other instruments are already in AAPP format

The satellite identifier is extracted from the input file name, so it must be in standard CLASS format, e.g. NSS.HIRX.N[A-P].D?????.\*. If the input file includes an archive header, this is automatically detected and removed.

For GAC datasets, the GAC line spacing is retained (1 line per 3 instrument scans, i.e. 2 lines per second). Across track, the 409 GAC spots are fitted into 2048 output spots.

The following executables are called, depending on instrument: avhrr\_gac\_class\_to\_aapp\_klm.exe, avhrr\_lac\_class\_to\_aapp\_klm.exe, msu\_class\_to\_aapp.exe, avhrr\_gac\_class\_to\_aapp\_a-j.exe, avhrr\_lac\_class\_to\_aapp\_a-j.exe, hirs2\_class\_to\_aapp.exe.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

Note: Prior to AAPP v7.6, a program **hrpt1b\_noaa** was used for AVHRR. This program is now obsolete, but its description is included here for completeness: **hrpt1b\_noaa.exe** opens the AVHRR NOAA level 1b file and the new AVHRR AAPP 11a/11b file (named **hrpt.11b**). Reading the AVHRR level 1b file record by record, the first 22016 bytes of each NOAA record (22528 bytes) are written in the AAPP file. To get information in the format and with the scaling factors expected by AAPP, it was necessary to run **avhrrcl** after getting the AVHRR AAPP 11a/11b file.

#### **4.1.26. Convert AVHRR 11b in AAPP format to NOAA format: **avhrr\_aapp\_to\_class** script and **avhrr\_aapp\_to\_class.exe****

Introduced in AAPP v7.6. This tool converts AVHRR level 1B in AAPP format to NOAA 16-bit (KLM) format.

#### **4.1.27. Initialisation before OPS-LRS software: **SATPOS-SVM.KSH, SATPOS-SVM.PL****

This module is used for creating a SVM file for OPS, using a **satpos** file as input. **Satpos** file contain indication on the exposition of the satellite to the sun, and these informations are transcribed in the SVM file. Note that only **UMBRA\_END** and **UMBRA\_START** informations are actually filled in the SVM file.

#### **4.1.28. Initialisation before OPS-LRS software: **MESSAGES-OSV.KSH, MESSAGES-OSV.PL****

This module is used for creating a OSV file for OPS, from messages extracted from the ASCII ADMIN buffer. Only messages reporting manoeuvres are actually transcribed to the OSV file.

#### **4.1.29. Navigation tools: **SATEPH** script, **LGEPHEING** script and **LGEPHING.EXE, LGEPHE** script and **LGEPHE.EXE, ALLEPH** script and **EPHE, TRACKING, ANTCNFT, DRIFTEPHE, TBUSDISP** script, **TBUSDISP.EXE, TLEPRINT** script, **TLEPRINT.EXE.****

Those modules are not called by the script **AAPP\_RUN\_NOAA**.

#### ***Module SATEPH***

---

(See also reference manual pages: *satpos.5, ephe.5*)

**sateph** module prepares a satellite position-velocity (**satpos**) file and an ephemeris (**ephe**) file for a given satellite and date. This module is of high interest and does some similar work as **alleph**, but the major purpose is that this module concerns only one satellite and creates the core navigation files for a given date, while **alleph** creates also the tracking files.

**Sateph** should be started before any new pass or once a day. Suggestion is to start **sateph** between the series of passes (for a local station) in order to get benefit from the newest orbital elements (retrieved by **get\_tle** or any similar tool); also start **sateph** before midnight for the next day (ie: *sateph -s noaa18 -d 1*)

**sateph** stores the outputs in the AAPP operational environment, **satpos** files in `${DIR_NAVIGATION}/satpos` and ephemeris files in `${DIR_NAVIGATION}/ephe`

#### **TASK 1: INPUT PARAMETER READING**

It gets:

- input command line parameters (satellite, bulletin type, date, number of days, station name ...)

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

- if bulletin type is missing it search in the global variable

#### TASK 2: INITIALISATION

If bulletin type is missing it search in the global variable PAR\_NAVIGATION\_DEFAULT\_LISTEBUL the corresponding bulletin type for the satellite.

From bulletin type it defines

- the name of the satpos command (**satpost** for TBUS, **satpostle** for 2-Line, **spatposspm** for SPOT)
- the file name for the bulletin index

#### TASK 3: RESULTS

For each satellite of the list, **sateph** :

- execute the satpos command (see above) and stores the result in the “operational environment” with file name: \${DIR\_NAVIGATION}/satpos/satpos\_*ssss\_yyyyymmdd.txt*
- execute the ephe command and stores the result in the “operational environment” with file name: \${DIR\_NAVIGATION}/ephe/ephe\_*ssss\_yyyyymmdd.txt*

#### Modules LGEPHEING, LGEPHEING.EXE

(See also reference manual pages: *lgepheing.1, lgephe.5, ephe.5*)

They are navigation tools useful to ingest a TBUS bulletin for long term ephemeris calculations.

**lgepheing** opens or creates an historical ephemeris utilities file required by the ephemeris files (long-term), into which new informations included in TBUS bulletin will be inserted. For each satellite of the list, orbital parameters useful for the ephemeris calculation will be extracted from the TBUS bulletin. The user chooses files relative to the considered satellites (input configuration).

#### TASK 1: INPUT PARAMETER READING

It gets:

- home directory of the TBUS files and the short name of the TBUS file
- the satellite list
- the historical ephemeris utilities file name

#### TASK 2: INITIALISATION

It opens the TBUS bulletin to process.

#### TASK 3: HISTORICAL FILES UPDATING

For each satellite of the list, **lgepheing** :

- opens (or creates) historical files
- extracts useful parameters for TBUS part IV and checks that the extracted parameters are in the authorised value area.(**tb\_dc**).
- writes a record in the historical ephemeris utilities file (**lge\_wind**)

#### Modules LGEPHE, LGEPHE.EXE

(See also reference manual pages *lgephe.1, lgephe.5, ephe.5*)

They are navigation tools useful to produce an ephemeris file, which contains the times of the ascending and descending nodes, the times of start and end of acquisition. **lgephe** produces a long term ephemeris file, i.e. over several months , for one satellite (due to the historical ephemeris utilities file) and several stations. In this case, the satellite position is calculated by an

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

approximate method assuming a circular orbit with linear variation of the nodal period and of the node longitude increment.

#### TASK 1: INPUT PARAMETER READING

It gets:

- the satellite name
- the start date and the number of days
- the historical ephemeris utilities file name
- the station names

#### TASK 2 : INITIALISATION

It opens the historical ephemeris utilities file and reads it by calling **lg\_gelem**.

**lg\_gelem** reads parameters preceding the stop time for the ephemeris in the historical ephemeris utilities file. It stores them in circular arrays of 30 elements (to be adapted according to the long term ephemeris duration). The stored values are used to compute linear regressions on the nodal period and longitude increment. Then it calculates the reference orbit that will be used for ephemeris calculations. This orbit contains the start time for the ephemeris.

It calls **gstatic** that gets the station coordinates (lat., lon., alt.) from the file *stations.txt* and then converts them into Greenwich cartesian coordinates.

#### TASK 3 : POSITIONS CALCULATION FOR ALL THE GIVEN TIME PERIOD

**lgephe** calls **lge\_ephe** to calculate the ascending and descending node times, starting and ending acquisition time for each station of the list. Information are stored into the long term ephemeris file.

**lge\_ephe** calculates (loop on every orbit from the reference orbit) the ascending and descending node times. Then, every orbit is cut out in calculation interval [t1,t2] with *tstep* duration, and we test for each station (loop on station) if there is a starting or ending time included in this interval. In that case, time and transition nature (starting or ending time into the reception area) is precisely determined.

To manage those tasks, **lge\_ephe** calls subroutines :

**satpoc** calculates satellite position for each calculation step according to a circular orbit.

**trackang** calculates satellite position in local station coordinates, then test if the satellite comes into or leaves a station area (loop on stations).

**instatic** calculates (with a dichotomic method) starting or ending acquisition time into considered time interval, assuming circular orbit (loop on stations).

**wephmes** is called each time different ephemeris messages had to be written into the output file.

#### Module ALLEPH

---

(See also reference manual pages: *alleph.1*)

**allephe** is the script that allows to run the NOAA ephemeris scheme for the short term. It runs for one acquisition station and loops on a satellite list. **allephe** calls **satpost.exe** (or **satpostle.exe** or **satposspm.exe** or **satposa.exe**) to create the satpos file (see above), **ephe.exe** for ephemeris, **tracking.exe** to compute tracking angles. Then mixes the satellites and identifies antenna conflicts by calling (**antcnft.exe,driftephe.exe**).

#### Module EPHE, EPHE.EXE

---

(See also reference manual pages: *ephe.1, lgephe.5, ephe.5*)

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**ephe** is a navigation tool useful to produce an ephemeris file, which contains the times of the ascending and descending nodes, the times of start and end of acquisition. It produces a short term ephemeris file, corresponding to duration of the input SATPOS file, which is relative to one satellite and one station.

To do this, **ephe** calls a main subroutine **sp\_ephe**.

#### TASK 1: INITIALISATION

**sp\_ephe** reads the header of the SATPOS file, checks whether the input period of time is included in the SATPOS period of time (if not the error flag *ierr* it set to 1) and determines the position-velocity number of calculation steps to read.

**sp\_ephe** calls **intrack** to calculate station values useful for the tracking angles calculations. The station is known in Greenwich reference frame by its geographic coordinates latitude, longitude, altitude. The viewing vector must be in the station local reference frame (zenith, south, east). So a transformation matrix from Greenwich to local reference frame has to be computed.

#### TASK 2: EPHEMERIS CALCULATION FOR THE GIVEN TIME PERIOD

**sp\_ephe** calculates times of various events as :

- sunset and sunrise for the station
- ascending and descending nodes
- station acquisition starting and ending
- maximum elevation during the pass

Ephemeris are calculated for each position-velocity read in SATPOS (loop). Information is stored in the ephemeris output file.

To manage those tasks, **sp\_ephe** calls subroutines :

**sunriset** calculates station sunrise and sunset times (depends on sun elevation angle) for a given day. So **sunriset** is called only once a day. Day test (to know if the day has changed) is made for each position-velocity read in SATPOS.

**intnode** calculates for a given time period (which must include equatorial pass), the relative time of the ascending or descending node from the satellite position-velocity for both limiting times of the period. The time where the z component of the satellite position is null, is determined with an iterative method, for which satellite position and velocity are calculated using a cubic interpolation. Since the node time is known, the satellite position-velocity is determined for that time and position is converted in longitude.

**intstat** calculates for a given time period, the relative time of the start and end of station acquisition from the satellite position-velocity for both limiting times of the period. Time where the satellite elevation angle from the station null is determined with an iterative method, for which the satellite position and velocity are calculated using a cubic interpolation. Since the acquisition time is known, satellite positions are determined for that time. Then, it is possible to deduce if the satellite came in or out of the station acquisition area.

**wephmes** is called each time different ephemeris messages must be written into the output file.

#### Module TRACKING, TRACKING.EXE

(See also reference manual pages: *tracking.1, tracking.5, lgephe.5, ephe.5*).

**tracking** is a navigation tool useful to produce the antenna tracking angle files corresponding to a satpos file. An antenna tracking angle file is produced for each orbit which is acquired by the station (even short acquisition). It contains the site (including a refraction correction) and azimuth values. The time step for calculations is an integer value expressed in seconds. It is defined as a data statement in the main program.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

Note: The run is done for only one satellite due to the satpos file

#### TASK 1: INITIALISATION

**tracking** calls **sp\_read** to read the SATPOS file between the start and end julian instants. If the start time equals 0, all file is processed.

#### TASK 2: CALCULATION OF THE ANTENNA TRACKING ANGLES

**tracking** calls **sp\_track** to do this task:

**sp\_track** begins to call **initrack** to calculate station values useful for the tracking angles calculations (see in this paragraph ephe/task1).

Then it tests if there is a new acquisition.

If a new acquisition is found, it computes the start of acquisition (**instat**, see in this paragraph ephe/task2). While the site is higher than a threshold, it computes the tracking angles (**intposvel** and **trackang** (see in this paragraph lgephe/task 3)). The sun position is calculated in Greenwich reference frame (**sungrw**).

**sp\_track** calls **wephmes** (see in this paragraph lgephe/task 3).

#### Module ANTCNFT, ANTCNFT.EXE

(See also reference manual pages *antcnft.1*, *ephe.5*)

**antcnft** (ANTenna CoNFlicT) identifies the acquisition conflicts for a single antenna system. It processes an ephemeris file which contains several satellites and is valid for only one station.

The ephemeris file for each satellite has been produced by ephe and tracking, and the various files have been concatenated and the final file has been sorted to be strictly chronological.

**antcnft** modifies this file to identify the orbits which are considered as conflict orbits.

A priority rule is established for the list of satellites read on unit 10, the first one having the higher priority, the second the following... . When several orbits are overlapping the orbit with the higher priority is kept and the other ones are identified as conflict orbits. The orbit duration is not taken into account. No margin is considered to identify overlapping orbits. For conflict orbits the event field of the ephemeris message becomes "start\_conflict" or "stop\_conflict".

#### Module DRIFTEPHE, DRIFTEPHE.EXE

It drifts the time of start of acquisition for a number of seconds.

#### Modules TBUSDISP, TBUSDISP.EXE

(See also reference manual pages: *tbusdisp.1*)

**tbusdisp.exe** displays the content of a TBUS file for any satellite by calling **tb\_gnv** that gets the nearest valid tbus filename from the index file, **tb\_glpv** that gets the last preceding valid tbus filename from the index file, **tb\_dc** that decodes the TBUS Part IV orbital elements, **clkerr\_dc** that decodes the clock error values stored in the plain language message at the end of the TBUS Part IV.

#### Modules TLEPRINT, TLEPRINT.EXE

(See also reference manual page: *tleprint.1*)

**tleprint.exe** displays the content of a Two-Line file for any satellite by calling **tle\_dc** that decodes the TLE orbital elements

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.1.30. BUFR tools

These tools allow the decoding and encoding of BUFR level 1c data for various instruments. BUFR format is used by EUMETSAT in their dissemination of global and regional ATOVS data and by NOAA in their distribution of global Suomi-NPP data. To use the tools, one of the ECMWF BUFR libraries must be installed (see AAPP Installation Guide): either BUFRDC or ecCodes.

##### Modules *AAPP\_DECODEBUFR\_1C, AAPP\_DECODEBUFR\_1C.EXE*

**aapp\_decodebufr** can process either a single file or a list of files, using the BUFRDC library.

The output file name is constructed from the input file name, with the suffix changed to “.11c”.

The following environment variable is required:

BUFR\_TABLES – directory containing BUFR tables (required)

usage : aapp\_decodebufr\_1c [-i files] [-v] [instruments]

where files is a list of files to decode. Quotes " " are necessary if there is more than 1 file.

**aapp\_decodebufr** calls **aapp\_decodebufr\_1c.exe** for each input file. It performs the following steps:

1. Opens the BUFR file
2. Reads each message and decodes it
3. For each message, examines the first word in the BUFR sequence to determine which instrument it contains
4. Calls subroutine **AAPP\_GET\_1C** to transfer the data to the AAPP 1c data structures  
**AAPP\_GET\_1C** calls different subroutines specific to each instrument :  
**aapp\_get\_1c\_XXX.F with XXX =amsua, amsub, msu, atms, hirs, iasi, pciasi, cris, mwts, mwts2, mwhs, mwhs2, iras.**
5. On conclusion it updates the 1c header, writes to disk and closes all files

##### Modules *ECCODES\_DECODEBUFR\_1C, ECCODES\_DECODEBUFR\_1C.EXE*

**eccodes\_decodebufr\_1c** decodes BUFR files using the ecCodes library.

The output file name is constructed from the input file name, with the suffix changed to “.11c”.

usage : eccodes\_decodebufr\_1c [-i files] [-v] [instruments]

where files is a list of files to decode. Quotes " " are necessary if there is more than 1 file.

**eccodes\_decodebufr\_1c** calls **eccodes\_decodebufr\_1c.exe** for each input file. It performs the following steps:

1. Opens the BUFR file
2. Reads each message and decodes it
3. For each message, examines the first word in the BUFR sequence to determine which instrument it contains
4. Calls subroutine specific to each instrument : **eccodes\_get\_1c\_XXX.F with XXX =amsua, amsub, msu, atms, hirs, iasi, pciasi, cris, crisfsr, mwts, mwts2, mwhs, mwhs2, iras.**
5. On conclusion it updates the 1c header, writes to disk and closes all files



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

***Modules AAPP\_ENCODEBUFR\_1C, AAPP\_ENCODEBUFR\_1C.EXE***

**aapp\_encodebufr\_1c** can process either a single file or a list of files, files in AAPP 11c format or files in AAPP 11d format, using the BUFRDC library.

The output file name is constructed from the input file name, with the suffix changed to “.bufr”.

The script requires as arguments a list of instrument types corresponding to the input files (i.e. HIRS, AMSU-A, AMSU-B, MHS, IASI, PCIASI, CRIS, ATMS, CRIS1D, ATMS1D, HIRS1D, AMSUB1D, IASI1D, MWTS MWHS IRAS MWTS2 MWHS2 MWTS21D, MWHS21D). A list of input file names may also be supplied (otherwise it assumes defaults hrsn.11c, aman.11c, ambn.11c, etc.)

The following environment variables may be used to define more precisely the encoding:

BUFR\_TABLES – directory containing BUFR tables (required)

ORIGINATING\_CENTRE – for Section 1 (default 254=EUMETSAT, or 74 for level 1d)

SUB\_CENTRE – for Section 1 (default 0)

MESSAGE\_SUBTYPE – locally defined subtype for section 1 (defaults vary with instrument)

MASTER\_TABLE – version number of master table (default 14, or 16 for ATMS/CrIS, or 26 if encoding of NEDT is required (ATOVS only))

LOCAL\_TABLE – version number of local table (default 0, or 1 for level 1d)

CENTRE\_ID – 1b/1c data originating centre, for section 4 (default 254=EUMETSAT)

SUB\_CENTRE\_ID – 1b/1c data originating sub-centre, for section 4 (default 0)

BUFR\_EDITION – BUFR edition number (default 4)

ENHANCED\_IASI – set this to Y to use the “day 2” IASI sequence 3-40-007, or N for 3-40-001. Default is Y.

ATMS\_THIN – (default 1) used to thin ATMS to 1 spot in *n* and 1 line in *n* in the BUFR output.

MWTS2\_THIN – (default 1) used to thin MWTS2 to 1 spot in *n* and 1 line in *n* in the BUFR output

MWHS2\_THIN – (default 1) used to thin MWHS2 to 1 spot in *n* and 1 line in *n* in the BUFR output

IRAS\_THIN – (default 1) used to thin IRAS to 1 spot in *n* and 1 line in *n* in the BUFR output

*Note:* if ATMS\_THIN, MWTS2\_THIN, etc. is set to a negative value then thinning is only performed in the along-scan direction; every scan will be output.

USE\_OB\_TIME – set this to Y to set the time stamp in Section 1 to the time of the first observation, or N to use the system time when the program is run. Default Y.

**aapp\_encodebufr\_1c** script calls **aapp\_encodebufr\_1c.exe** for each instrument. It performs the following steps:

1. Defines the BUFR sequence descriptor(s) for the required instrument
2. Sets up the fixed parts of the message
3. Calls subroutine **AAPP\_PUT\_1C** to open the input file, read records into AAPP structures and copy data to the “VALUES” array  
**AAPP\_PUT\_1C** calls different subroutines specific to each instrument :  
**aapp\_put\_1c\_XXX.F** with **XXX =amsua, amsub, msu, atms, hirs, iasi, pciasi, cris, etc.**
4. Encode each message and write to output file
5. On conclusion, close all files

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

### Modules *ECCODES\_ENCODEBUFR\_1C*, *ECCODES\_ENCODEBUFR\_1C.EXE*

**eccodes\_encodebufr\_1c** can process either a single file or a list of files, files in AAPP 11c format or files in AAPP 11d format, using the ecCodes library.

The output file name is constructed from the input file name, with the suffix changed to “.bufr”.

The script requires as arguments a list of instrument types corresponding to the input files (i.e. HIRS, AMSU-A, AMSU-B, MHS, IASI, PCIASI, CRIS, CRISFSR, ATMS, CRIS1D, ATMS1D, HIRS1D, AMSUB1D, IASI1D, MWTS MWHS IRAS MWTS2 MWHS2 MWTS21D, MWHS21D). A list of input file names may also be supplied (otherwise it assumes defaults hrsn.11c, aman.11c, ambn.11c, etc.)

The following environment variables may be used to define more precisely the encoding:

ORIGINATING\_CENTRE – for Section 1 (default 254=EUMETSAT)

SUB\_CENTRE – for Section 1 (default 0)

MESSAGE\_SUBTYPE – locally defined subtype for section 1 (defaults vary with instrument)

MASTER\_TABLE – version number of master table (default 16, or 26 if NEDT is required (ATOVS only))

LOCAL\_TABLE – version number of local table (default 0, or 1 for level 1d)

CENTRE\_ID – 1b/1c data originating centre, for section 4 (default 254=EUMETSAT)

SUB\_CENTRE\_ID – 1b/1c data originating sub-centre, for section 4 (default 0)

MESSAGES\_PER\_SCAN\_XXX where xxx can be IASI, PCIASI, CRIS, CRISFSR – can be used to reproduce the behavior of AAPP v7 (default for v8 is 1 message per scan).

**eccodes\_encodebufr\_1c** script calls **eccodes\_encodebufr\_1c.exe** for each instrument. It performs the following steps:

1. Defines the BUFR sequence descriptor(s) for the required instrument
2. Sets up the fixed parts of the message
3. Calls subroutines specific to each instrument : **eccodes\_put\_1c\_XXX.F with XXX =amsua, amsub, msu, atms, hirs, iasi, pciasi, cris, etc.**
4. Encode each message and write to output file
5. On conclusion, close all files

#### 4.1.31. HDF5 tools

##### **cris\_sdr, cris\_sdr.exe**

Converts Sensor Data Record (SDR) in HDF5 to AAPP internal binary format and applies apodization.

Usage: **cris\_sdr** [-o Outputfile] [-g Geofile] [-H] [-B] [-N] SDRfile

Default apodization is Hamming (-H); alternatives are Blackman-Harris (-B) or none (-N). If the geolocation file (Geofile) is not specified in the command then the program attempts to read the geolocation file specified in the SDR.

Note: the maximum number of granules expected in an SDR, and the number of scans per granule, are defined in **cris\_sdr.h** (for C code) and also in **cris\_sdr\_out.F**. These may need to be changed to suit the incoming data.

##### **atms\_sdr, atms\_sdr.exe**

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

Converts ATMS Sensor Data Record (SDR) in HDF5 to AAPP internal binary format.

usage: `atms_sdr [-o Outputfile] SDRfile [TDRfile]`

Note: the maximum number of granules expected in an SDR, and the number of scans per granule, are defined in `atms_sdr.h` (for C code) and also in `atms_sdr_out.F`. These may need to be changed to suit the incoming data.

#### **avh1b\_to\_hdf5, avh1b\_to\_hdf5.exe**

Converts AVHRR level 1b AAPP format to HDF5.

Usage: `avh1b_to_hdf.exe infile outfile`

Read the whole AVHRR 1b file into memory. Convert raw counts to scaled radiance and reflectivities. Write out as HDF5.

#### **mwts\_sdr, mwts\_sdr.exe**

Converts MWTS SDR files in HDF-5 format to AAPP 1c format

This program ingests SDR files for the Microwave Temperature Sounder (MWTS) instrument on the Chinese FY-3 series. It converts from HDF5 to AAPP binary format (specified in “include” file `mwts.h`). The early releases of MWTS data by CMA suffered from limited quality control of the brightness temperatures and geolocation, therefore AAPP performs the following additional QC checks:

- . Calibration slope: reject scans having a slope less than 99% of the median
- . Latitude/longitude check: scan to scan consistency and difference across the scan
- . Reject scans with lunar contamination in space view

usage: `mwts_sdr [-o Outputfile] SDRfile`

If `Outputfile` is not specified the name of the output file is the same as the name of the input file, except that the suffix is changed to “.11c”.

#### **mwts2\_sdr, mwts2\_sdr.exe**

Converts MWTS2 SDR files in HDF-5 format to AAPP 1c format. Usage is as above, but for the MWTS2 instrument on FY-3C and later satellites. Quality control is limited to checking the geolocation.

#### **mwhs\_sdr, mwhs\_sdr.exe**

Converts MWHS SDR files in HDF-5 format to AAPP 1c format

This program ingests SDR files for the Microwave Temperature Sounder (MWHS) instrument on the Chinese FY-3 series. It converts from HDF5 to AAPP binary format (specified in “include” file `mwhs.h`). The early releases of MWHS data by CMA suffered from limited quality control of the brightness temperatures and geolocation, therefore AAPP performs the following additional QC checks:

- . Space and black body viewing angles: reject scans with errors greater than 100 counts compared with nominal positions
- . Check consistency of different time stamps within the dataset
- . Latitude/longitude check: scan to scan consistency and difference across the scan
- . Reject scans with lunar contamination in space view

usage: `mwhs_sdr [-o Outputfile] SDRfile`

If `Outputfile` is not specified the name of the output file is the same as the name of the input file, except that the suffix is changed to “.11c”

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### **mwhs2\_sdr, mwhs2\_sdr.exe**

Converts MWHS2 SDR files in HDF-5 format to AAPP 1c format. Usage is as above, but for the MWTS2 instrument on FY-3C and later satellites. Quality control is limited to checking the geolocation.

#### **iras\_sdr, iras\_sdr.exe**

Convert IRAS SDR files in HDF-5 format to AAPP 1c format.

#### **hiras\_sdr, hiras\_sdr.exe**

Convert HIRAS SDR files in HDF-5 format to AAPP 1c format. HIRAS is the hyperspectral sounder on FY-3D and later satellites.

#### **convert\_to\_hdf5**

Convert an AAPP-format level 1c file, or AVHRR 1b, to hdf5 format. The hdf5 structure closely mirrors that of the 11c files. It is *not* the same as the SDR files mentioned in the previous paragraphs.

The script calls various executables depending on instrument, e.g. ama1c\_to\_hdf5.exe, amb1c\_to\_hdf5.exe, atms1c\_to\_hdf5.exe, avh1c\_to\_hdf5.exe, hrp1b\_to\_hdf5.exe, etc.

#### **convert\_from\_hdf5**

This is the inverse of *convert\_to\_hdf5* described above, i.e. it converts from hdf5 back to AAPP level 1c.

The script calls executables such as ama1c\_from\_hdf5.exe, amb1c\_from\_hdf5.exe, etc.

#### **aapp1c\_quicklook.py**

A Python script to display the hdf5 files (11c.h5 or 11d.h5) from sounder instruments (AMSU, AMSU, MHS, HIRS, ATMS, MWHS2, MWTS2, IRAS, CrIS, IASI). Pre-requisites are numpy, matplotlib, h5py, cartopy. See section 4.3.48 for details.

## **4.2. INTERFACES**

Formats are detailed in the NWPSAF-MF-UD-003 (AAPP documentation/data formats)

For the input options and arguments, see the paragraph 4.3 “dynamic articulation”.

### **4.2.1. User input parameters in ATOVS\_ENV/ATOVS\_ENV8**

In AAPP versions 1 to 5 the ATOVS\_ENV file was located in the user’s home directory. For AAPP version 6, 7 or 8, ATOVS\_ENV is now called ATOVS\_ENV6 (or 7 or 8) and it is by default located in the installation top directory. This makes it easier to run different versions of AAPP on the same computer. However the built-in scripts do not source the ATOVS\_ENV8 file directly, they source a file ATOVS\_CONF which tests to see whether an ATOVS\_ENV8 file exists in the users’s home directory. If one does exist it will be used; if it does not exist the ATOVS\_ENV8 file in the installation top directory will be used. The user may customize ATOVS\_CONF if necessary to modify this behaviour.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

The ATOVS\_ENV8 file defines several environment variables. The user has to ensure of the set-up of the different variables.

The text, that follows, can make reference to those variables.

#### 4.2.2. Inputs/outputs for TBUSING navigation initialisation

##### *Inputs*

---

##### **TBUS\_YYYYMMDD.TXT**

TBUS bulletin, yyyy(year) mm(month) dd(day).

Located in the directory \${DIR\_NAVIGATION}/tbus\_db or orb\_elem/yyyy-mm.

yyyymmdd is the date of transmission of the bulletin by NOAA.

Bulletins are classified by year and month of transmission.

More details are given in *tbus.5*.

##### *Outputs :*

---

##### **TBUS\_NOAAXX.INDEX**

Historical TBUS index file for orbital parameters associated with a specific satellite, xx satellite number.

Located in the directory \${DIR\_NAVIGATION}/tbus\_db or orb\_elem.

The first line (header line) contains the NOAA name of the satellite.

Each following line contains epoch time in the CNES julian days (day 0=01/01/50 00h), quality flag (zero is good data), orbit number, extrapolation errors of position (km/day, 2 values forward and backward), the time-string (dd/mm/yy hh:mm:ss.sss), and the name of the TBUS file (full name).

More details are given in *tbus.5*.

##### **CLKERR\_NOAAXX.TXT**

Clock drift data file (ASCII) for each satellite, xx satellite number

Located in the directory \${DIR\_NAVIGATION}/tbus\_db or orb\_elem.

The first line (header line) contains the NOAA name of the satellite.

The second line has the name of the fields present in the following lines.

Each data line contains : an identification code (cerr, last, next, rate and plus bias for NOAA16), the date in CNES Julian days (day 0=01/01/50 00h), the value of cerr or last (in seconds) or rate (in ms/day) or bias (in seconds).

More details are given in *clockerror.5*.

##### **SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Named **tbusing.log**

The commands "print\*" , "write(\*, )" and the calls to subroutines ml\_wt.. write into it.

#### 4.2.3. Inputs/outputs for GET\_TLE navigation initialization

**get\_tle** retrieves the 2-Line orbital elements from a web site.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

***Inputs :***

---

**WEB SITE FOR 2-LINE ELEMENTS**

- URL, user, password are defined in the ATOVS\_ENV8 parameter file

***Outputs :***

---

**TLE\_YYYYMMDD\_HHMN.TXT**

- 2-Line elements retrieved on *yyyymmdd* at *hh:mn*

**4.2.4. Inputs/outputs for GET\_TAI\_UT1.UTC navigation tool**

**get\_tai\_ut1\_utc** retrieves time conversion and polar motion values from a reference web site

***Inputs :***

---

**WEB SITE FOR TAI, UI-UTC AND POLAR MOTION**

- URLs are defined in the ATOVS\_ENV6 or ATOVS\_ENV7 parameter file

***Outputs :***

---

**FINALS2000A.DATA**

- Polar motion and UTC-UT1 values (observed and forecast) stored under directory *\$DIR\_DATA\_TAI\_UT1.UTC*

**TAI-UTC.DAT**

- TAI UTC time difference, stored under *\$DIR\_DATA\_TAI\_UT1.UTC*

**4.2.5. Inputs/outputs for TLEING navigation initialisation**

***Inputs***

---

**TLE\_YYYYMMDD\_HHMN.TXT**

TLE bulletin, *yyyy*(year) *mm*(month) *dd*(day) *hh* (hour) *mn* (minute).  
 Located in the directory *\${DIR\_NAVIGATION}/tle\_db* or *orb\_elem/yyyy-mm*.  
*yyyymmdd hhmn* is the date and time of reception of the bulletin.  
 Bulletins are classified by year and month of reception.  
 More details are given in *tle.5* .

***Outputs :***

---

**TLE\_NOAAXX.INDEX**

Historical TLE index file for orbital parameters associated with a specific satellite, *xx* satellite number.  
 Located in the directory *\${DIR\_NAVIGATION}/tle\_db* or *orb\_elem*.  
 The first line (header line) contains the NOAA name of the satellite.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

Each following line contains epoch time in the CNES julian days (day 0=01/01/50 00h), quality flag (zero is good data), orbit number, extrapolation errors of position (km/day, 2 values forward and backward), the time-string (dd/mm/yy hh:mm:ss.sss), and the name of the TLE file (full name). More details are given in *tle.5*.

#### **SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Standard error output

The commands “print\*”, “write(\*, )” and the calls to subroutines ml\_wt.. write into it.

#### **4.2.6. Inputs/outputs for SATPOST navigation initialisation**

##### ***Inputs :***

---

#### **TBUS\_YYYYMMDD.TXT**

See input of tusing

#### **TBUS\_NOAAXX.INDEX**

See output of tusing

#### **STATIONS.TXT**

ASCII file containing geographic coordinates of reception station

Located in the directory \${DIR\_STATIONS}/stations.txt.

Each line contains the following information : latitude(deg)/longitude(deg)/altitude(km), elevation min. (deg) and name.

##### ***Outputs :***

---

#### **SATPOS\_NOAAXX\_YYYYMMDD.TXT**

Satellite position-velocity ASCII file associated with a given station and a given satellite, xx (satellite number) yyyy(year) mm(month) dd(day).

Located in the directory \${DIR\_NAVIGATION}/satpos.

Some dummy lines may exists at the beginning of the file. A line with the string #satpos indicates the actual beginning of the file.

The file header contains following information: names of satellite and station, start date, number of day, calculation time step, type, research criteria of the orbital bulletin and name of orbital bulletin, orbital parameters (date, semi-major axis (km), eccentricity, inclination (deg), perigee argument (deg), right ascension (deg), mean anomaly (deg), x,y,z positions (km), vx,vy,vz velocities (km/s)), ground station coordinates (latitude/longitude (deg), altitude (km), min. visibility (deg)).

Each data line contains : step number, position vector, inertial velocity vector, orbit number, satellite in daylight (0) or night-time (1) conditions, satellite seen from the station (0=yes, 1=no).

More details are given in *satpos.5*.

#### **SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Named **satpost.log**

The commands “print\*”, “write(\*, )” and the calls to subroutines ml\_wt.. write into it.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.2.7. Inputs/outputs for SATPOSTLE navigation initialisation

##### *Inputs :*

---

##### **TLE\_YYYYMMDD\_HHMN.TXT**

See input of tleing

##### **TLE\_NOAAXX.INDEX**

See output of tleing

##### **STATIONS.TXT**

ASCII file containing geographic coordinates of reception station

Located in the directory \${DIR\_STATIONS}/stations.txt.

Each line contains the following information : latitude(deg)/longitude(deg)/altitude(km), elevation min. (deg) and name.

##### *Outputs :*

---

##### **SATPOS\_NOAAXX\_YYYYMMDD.TXT**

Satellite position-velocity ASCII file associated with a given station and a given satellite, xx (satellite number) yyyy(year) mm(month) dd(day).

Located in the directory \${DIR\_NAVIGATION}/satpos.

Some dummy lines may exists at the beginning of the file. A line with the string #satpos indicates the actual beginning of the file.

The file header contains following information: names of satellite and station, start date, number of day, calculation time step, type, research criteria of the orbital bulletin and name of orbital bulletin, orbital parameters (date, semi-major axis (km), eccentricity, inclination (deg), perigee argument (deg), right ascension (deg), mean anomaly (deg), x,y,z positions (km), vx,vy,vz velocities (km/s)), ground station coordinates (latitude/longitude (deg), altitude (km), min. visibility (deg)).

Each data line contains : step number, position vector, inertial velocity vector, orbit number, satellite in daylight (0) or night-time (1) conditions, satellite seen from the station (0=yes, 1=no).

More details are given in *satpos.5*.

##### **SUMMARY FILE FOR PASS :**

Sequential file in ASCII text in standard output.

The commands "print\*" , "write(\*, )" and the calls to subroutines ml\_wt.. write into it.

#### 4.2.8. Inputs/outputs for decommutation (DECOMMUTATION)

##### *Inputs:*

---

##### **RAW DATA LEVEL0 :**

Unpacked HRPT minor frame(s) coming from the center specific module closely connected to the hardware.

The HRPT minor frame is an array of 11090 words, made of the 10 bits HRPT words right justified in 16 bit words.

Different informations are getting from this input file by calling the **hrptidf** program.



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

## AMSUA\_CLPARAMS.DAT

Sequential file in ASCII text.

Self-documented (lines of comments begin with "#").

Used for AMSU-A decommutation and AMSU-A calibration

There is one file for all the satellites with different sections for :

- AMSU\_A1 of NOAA15      ## AMSU-A1 FM1 DATA ## ## ID of instrument 9
- AMSU\_A2 of NOAA15      ## AMSU-A2 PFM DATA ## ## ID of instrument 6
- AMSU\_A1 of NOAA16      ## AMSU-A1 PFM DATA ## ## ID of instrument 5
- AMSU\_A2 of NOAA16      ## AMSU-A2 FM1 DATA ## ## ID of instrument 10
- AMSU\_A1 of NOAA17      ## AMSU-A1 FM2 DATA ## ## ID of instrument 13
- AMSU\_A2 of NOAA17      ## AMSU-A2 FM2 DATA ## ## ID of instrument 14
- AMSU\_A1 of NOAA18      ## AMSU-A1 FM3 DATA ## ## ID of instrument 33
- AMSU\_A2 of NOAA18      ## AMSU-A2 FM3 DATA ## ## ID of instrument 18
- AMSU\_A1 of NOAA19      ## AMSU-A1 DATA S/N 107 on NOAA-19 ##
- AMSU\_A2 of NOAA19      ## AMSU-A2 DATA S/N 109 on NOAA-19 ##
- AMSU\_A1 of METOP-A      ## AMSU-A1 S/N 106 on METOP-A ##
- AMSU\_A2 of METOP-A      ## AMSU-A2 S/N 108 on METOP-A ##
- AMSU\_A1 of METOP-B      ## AMSU-A1 S/N 108 on METOP-B ##
- AMSU\_A2 of METOP-B      ## AMSU-A2 S/N 106 on METOP-B ##
- Values for Fundamental Constants are common for all the satellites.

The file must be modified in the following cases:

- Insertion of the parameters of a new satellite (furnished just before the launch).

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 50 (see **decommutation.ksh**).

Located in the directory ../AAPP/src/calibration/libamsuacl and copied into the directory \${PAR\_CALIBRATION\_COEF}/amsua by the installation script.

### *Outputs:*

---

#### **LEVEL 1A DATA FILES :**

Direct access and unformatted binary files separated for each instrument according to the input options (one file for one instrument).

Files are named :

**hrrsn.11b msun.11b aman.11b ambn.11b dcsn.11b hrpt.11b**

Note that ambn.11b can contain either AMSU-B or MHS data, depending on the satellite. Files are renamed at the end of **AAPP\_RUN**

**hirsl1b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.11b**

**msul1b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.11b**

**amsual1b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.11b**

**amsubl1b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.11b**

**hrpt\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.11b**

with     SATIMG : satellite name (example noaa16)

          YYYYMMDD.: year-month-day of data

          HHMN : time of data

          NNNNN : orbit number

Each file contains: 1 header record +1 data record for each scan line

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

The size of the record depends on the instrument

- 22016 bytes for AVHRR (does not respect 1B NOAA size, see AAPP documentation/data formats))
- 4608 bytes for HIRS
- 2560 bytes for AMSU-A
- 3072 bytes for AMSU-B/MHS
- 1024 bytes for MSU
- 10752 bytes for DCS

Calibration and location fields are set to zero.

Each data record for a level 1a line contains counts + time + housekeeping information.

For the HIRS, AMSU-A, AMSU-B and MHS, the level 1a files are very close to the NOAA 1b formats. The differences are in some scaling factors.

For the MSU, AAPP has developed its own MSU.11b format. It is very close to the HIRS, AMSU-A and AMSU-B formats.

For the AVHRR, the file is different from NOAA one (see AAPP documentation/data formats).

For all the instruments, there are no missing lines (different from NOAA format)

To get the details of the files, see the corresponding include files.

Associated with logical units (see **decommutation.ksh**):

- 11 for **hrsn.11b**
- 12 for **msun.11b**
- 13 for **dcsn.11b**
- 14 for **hrpt.11b**
- 15 for **aman.11b**
- 16 for **ambn.11b**

Located in the directory `${WRK}`.

#### **SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Named **decommutation.log**

The commands “print\*”, “write(\*, )” and the calls to subroutines ml\_wt.. write into it.

Located in the directory `${WRK}`.

#### **4.2.9. Inputs/outputs EPS level 0 format to AAPP level 1a format**

##### ***Inputs:***

See documents [25]

##### ***Outputs:***

#### **LEVEL 1A DATA FILES :**

It is the same format that the Decommuration outputs.

Named : **hrsn.11b msun.11b aman.11b ambn.11b hrpt.11b**

File **ambn.11b** contains either AMSU-B or MHS data, depending on the satellite.

Outputs of the decommutation task.

Logical units used can differ following the instruments to process . See the corresponding scripts.

More often, the associated logical unit is 11.

Located in the directory `${WRK}`.

More details, see outputs of **decommutation**.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.2.10. Inputs/outputs ATOVS and AVHRR navigation (HIRSCL, HIRSCL\_ALGOV4, MSUCL, AMSUACL, AMSUBCL, MHSCCL, AVHRCL)

##### *Inputs :*

---

##### **LEVEL 1B DATA FILES :**

Named : **hrsn.l1b msun.l1b aman.l1b ambn.l1b hrpt.l1b**

File **ambn.l1b** contains either AMSU-B or MHS data, depending on the satellite.

Outputs of the decommutation task.

Logical units used can differ following the instruments to process . See the corresponding scripts.

More often, the associated logical unit is 11.

Located in the directory `${WRK}`.

More details, see outputs of **decommutation**.

##### **SATPOS\_NOAAXX\_YYYYMMDD.TXT**

ASCII file.

Satellite position-velocity associated with a given station and a given satellite with xx satellite number, yyyymmdd start date of position-velocity calculation.

Output of the **satpost** or **satpostle** command.

Logical unit used can differ following the instruments to process . See the corresponding scripts. More often, the associated logical unit is 15.

Located in the directory `${DIR_NAVIGATION}/satpos`.

More details are given in *satpos.5* .

##### **CLKERR\_NOAAXX.TXT**

ASCII file.

Historical clock error file associated with a specific satellite, xx satellite number

Output of the **tbusing** command.

Logical units used can differ following the instruments to process . See the corresponding scripts.

More often, the associated logical unit is 16.

Located in the directory `${DIR_NAVIGATION}/tbus_tb` or `orb_elem`.

More details are given in *clockerror.5* .

##### *Outputs :*

---

##### **LEVEL 1B DATA FILES :**

Files are named : **hrsn.l1b msun.l1b aman.l1b ambn.l1b hrpt.l1b**

Files are renamed at the end of **AAPP\_RUN**

**hirs11b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.l1b**

**msul1b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.l1b**

**amsual1b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.l1b**

**amsubl1b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.l1b**

**hrpt\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.l1b**

with `SATIMG` : satellite name (example noaa16)

`YYYYMMDD`.: year-month-day of data

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

HHMN : time of data

NNNNN : orbit number

Compared to level.1a structure, 'navigation' parameters have been updated.

Located in the directory \${WRK}.

More details, see outputs of **decommutation**

#### 4.2.11. Inputs/outputs HIRS calibration (first algorithm) (HIRSCL)

##### *Inputs :*

---

##### **HIRS LEVEL 1A DATA FILE :**

Named **hrsn.11b**.

Output of the decommutation task.

Associated with logical unit 11 (see **hirscl.ksh**).

Located in the directory \${WRK}.

More details, see outputs of **decommutation**.

##### **CALCOEF.DAT**

Sequential file in ASCII text.

Contains calibration HIRS parameters.

Self-documented (lines of comments begin with "#").

One file for all the satellites (with 1 section for each).

C1 and C2 constants, used in the function of Planck are the same for all the satellites.

This file must be modified in the following cases :

- Insertion of the parameters of a new satellite (furnished just before the satellite launch).
- When the range of values are too strict and excludes too many values (that's why sometime there is no calibration for a channel). So, modification of these values is needed. For example, lighting conditions of the satellite change according to the season. This phenomenon induces variations in the observed numerical counts (e.g. NOAA12 in May and September).

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 12 (see **hirscl.ksh**)

Located in the directory ../AAPP/src/calibration/libhirscl and copied into the directory \${PAR\_CALIBRATION\_COEF}/hirs by the installation script.

##### **TESTCOEF.DAT**

Sequential file in ASCII text

Contains the values of the parameters used in calibration tests.

Self-documented (lines of comments begin with "#").

Common values for all the satellites.

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 13 (see **hirscl.ksh**)

Located in the directory ../AAPP/src/calibration/libhirscl and copied into the directory \${PAR\_CALIBRATION\_COEF}/hirs by the installation script.

##### *Outputs :*

---

##### **HIRS LEVEL 1B DATA FILE :**

Named **hrsn.11b**.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

File is renamed at the end of **AAPP\_RUN**.

**hirs11b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.11b**

Compared to level.1a structure, 'calibration' parameters have been updated.

Associated with logical unit 11 (see **hirscl.ksh**).

Located in the directory \${WRK}.

More details, see outputs of **decommutation**.

**MONHIRS.TXT**

Formatted file in ASCII text.

Contains various statistics parameters showing the evolution of the calibration coefficient calculation.

Filled during the run of **hirscl.exe** if specified in input options. One record added for one run.

Associated with logical unit 14 (see **hirscl.ksh**).

Located in the directory \${PAR\_CALIBRATION\_MONITOR}/noaaXX with XX satellite number.

**SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Named **hirscl.log**

The commands “print\*”, “write(\*, )” and the calls to subroutines ml\_wt.. write into it.

Located in the directory \${WRK}

**4.2.12. Inputs/outputs HIRS calibration algorithm version 4 – part 1  
(HCALCB1\_ALGOV4)**

*Inputs :*

---

**HIRS\_HISTORIC.TXT**

Formatted file in ASCII text.

Contains values of various parameters used into the calculation of the calibration coefficients.

Filled during the run of **hirscl\_algoV4.exe** : 70 lines added for one qualified calibration cycle of an orbit.

If the file doesn't exist (after the launch of the satellite for example), the script **hcalcb1\_algoV4** create the file (empty file named **hirs\_historic**).

Located in the directory \${PAR\_CALIBRATION\_MONITOR}/noaaXX with XX satellite number.

*outputs :*

---

**HIRS\_BIASLOPE.TXT**

Sequential file in ASCII text of 22 lines

Contains the date and time of a reference time, the number of hours. The two parameters determines the period of the HIRS data used to compute the b1 coefficients and the average slopes. Contains the 19 b1 coefficients and the 19 average slopes.

Located in the directory \${WRK}.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.2.13. Inputs/outputs HIRS calibration algorithm version 4 – part 2 (HIRSCL\_ALGOV4)

##### *Inputs :*

---

##### **HIRS LEVEL 1A DATA FILE :**

Named **hrs\_n11b**.

Output of the decommutation task.

Associated with logical unit 11 (see **hirscl\_algoV4.ksh**).

Located in the directory `${WRK}`.

More details, see outputs of **decommutation**.

##### **HIRS\_BIASLOPE.TXT**

Sequential file in ASCII text of 22 lines

Contains the date and time of a reference time, the number of hours. The two parameters determines the period of the HIRS data used to compute the b1 coefficients and the average slopes. Contains the 19 b1 coefficients and the 19 average slopes.

Output of the script `hcalcb1_algoV4` that must run before **hirscl\_algoV4**.

Associated with logical unit 14 (see **hirscl\_algoV4.ksh**).

Located in the directory `${WRK}`.

##### **CALCOEF\_ALGOV4.DAT**

Sequential file in ASCII text.

Contains calibration HIRS parameters.

Self-documented (lines of comments begin with "#").

One file for all the satellites (with 1 section for each).

C1 and C2 constants, used in the function of Planck are the same for all the satellites.

This file must be modified in the following cases :

- Insertion of the parameters of a new satellite (furnished just before the satellite launch).
- When the range of values are too strict and excludes too many values (that's why sometime there is no calibration for a channel). So, modification of these values is needed. For example, lighting conditions of the satellite change according to the season. This phenomenon induces variations in the observed numerical counts (e.g. NOAA12 in May and September).

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 12 (see **hirscl\_algoV4.ksh**)

Located in the directory `../AAPP/src/calibration/libhirscl_algoV4` and copied into the directory `${PAR_CALIBRATION_COEF}/hirs` by the installation script.

##### **TESTCOEF\_ALGOV4.DAT**

Sequential file in ASCII text

Contains the values of the parameters used in calibration tests.

Self-documented (lines of comments begin with "#").

Common values for all the satellites.

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 13 (see **hirscl\_algoV4.ksh**)

Located in the directory `../AAPP/src/calibration/libhirscl_algoV4` and copied into the directory `${PAR_CALIBRATION_COEF}/hirs` by the installation script.

##### *Outputs :*

---

##### **HIRS LEVEL 1B DATA FILE :**

Named **hrs\_n11b**.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

File is renamed at the end of **AAPP\_RUN**.

**hirs11b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.11b**

Compared to level.1a structure, 'calibration' parameters have been updated.

Associated with logical unit 11 (see **hirscl\_algoV4.ksh**).

Located in the directory **\${WRK}**.

More details, see outputs of **decommutation**.

### **HIRS\_HISTORIC.TXT**

Formatted file in ASCII text.

Contains values of various parameters used into the calculation of the calibration coefficients and later used to compute the b1 coefficients and the average slopes.

70 lines added for one qualified calibration cycle of an orbit.

Can contain values for several orbit runs.

The script **hirs\_historic\_file\_manage.ksh** manages the file: When the file has a number of lines superior to a define number (see **hirs\_historic\_file\_manage.ksh**), it is copied to **hirs\_historic.txt.0** file. If **hirs\_historic.txt.0** file already exists, it is moved to **hirs\_historic.txt.1** . to **hirs\_historic.txt.max** can be stored (see **hirs\_historic\_file\_manage.ksh** for max). The final part of **\${HIST}** is remained in **\${HIST}**.

Associated with logical unit 15 (see **hirscl\_algoV4.ksh**).

Located in the directory **\${PAR\_CALIBRATION\_MONITOR}/noaaXX** with XX satellite number.

### **SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Named **hirscl.log**

The commands "print\*", "write(\*, )" and the calls to subroutines **ml\_wt..** write into it.

Located in the directory **\${WRK}**

## **4.2.14. Inputs/outputs MSU calibration (MSUCL)**

### ***Inputs :***

---

#### **MSU LEVEL 1A DATA FILE :**

Named **msun.11b**.

Output of the decommutation task.

Associated with logical unit 11 (see **msucl.ksh**).

Located in the directory **\${WRK}**.

More details, see outputs of **decommutation**.

#### **CALCOEF.DAT**

Sequential file in ASCII text.

Self-documented (lines of comments begin with "#").

Contains calibration MSU parameters.

One file for all the satellites (with 1 section for each).

C1 and C2 constants, used in the function of Planck are the same for all the satellites.

This file must be modified in the following cases :

- Insertion of the parameters of a new satellite (furnished just before the satellite launch).
- When the reference temperature is too far from the most computed temperatures. Messages are printed (see different examples in the comment section).

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 12 (see **msucl.ksh**).

Located in the directory ../AAPP/src/calibration/libmsucl and copied into the directory \${PAR\_CALIBRATION\_COEF}/msu by the installation script.

### **TESTCOEF.DAT**

Sequential file in ASCII text.

Contains the values of the parameters used in calibration tests.

Self-documented (lines of comments begin with "#").

Common values for all the satellites.

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 13 (see **msucl.ksh**)

Located in the directory ../AAPP/src/calibration/libmsucl and copied into the directory \${PAR\_CALIBRATION\_COEF}/msu by the installation script.

#### ***Outputs :***

---

### **MSU LEVEL 1B DATA FILE :**

Named **msun.l1b**

File is renamed at the end of **AAPP\_RUN**

**msul1b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.l1b**

Compared to level.1a structure, 'calibration' parameters have been updated.

Associated with logical unit 11 (see **msucl.ksh**).

Located in the directory \${WRK}.

More details, see outputs of **decommutation**.

### **MONMSU.TXT**

Formatted file in ASCII text

Contains various statistics parameters showing the evolution of the calibration coefficient calculation.

Filled during the run of **msucl.exe** if specified in input options. One record added for one run.

Associated with logical unit 14 (see **msucl.ksh**).

Located in the directory \${PAR\_CALIBRATION\_MONITOR}/noaaXX with XX satellite number.

### **SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Named **msucl.log**

The commands "print\*", "write(\*, )" and the calls to subroutines ml\_wt.. write into it.

Located in the directory \${WRK}

## **4.2.15. Inputs/outputs AMSU-A calibration (AMSUACL)**

#### ***Inputs :***

---

### **AMSU-A LEVEL 1A DATA FILE :**

Named **aman.l1b**.

Output of the decommutation task.

Associated with logical unit 11 (see **amsucl.ksh**).

Located in the directory \${WRK}.

More details, see outputs of **decommutation**.



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

### AMSUA\_CLPARAMS.DAT

Sequential file in ASCII text

Self-documented (lines of comments begin with "#").

Used for AMSU-A decommutation and AMSU-A calibration.

There is one file for all the satellites with different sections for :

- AMSU-A1 of NOAA15      ## AMSU-A1 FM1 DATA ## ## ID of instrument --> 9
- AMSU-A2 of NOAA15      ## AMSU-A2 PFM DATA ## ## ID of instrument --> 6
- AMSU-A1 of NOAA16      ## AMSU-A1 PFM DATA ## ## ID of instrument --> 5
- AMSU-A2 of NOAA16      ## AMSU-A2 FM1 DATA ## ## ID of instrument --> 10
- Values for Fundamental Constants are common for all the satellites.

This file must be modified in the following cases :

- Insertion of the parameters of a new satellite (furnished just before the satellite launch).

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 12 (see **amsuacl.ksh**).

Located in the directory ../AAPP/src/calibration/libamsuacl and copied into the directory \${PAR\_CALIBRATION\_COEF}/amsua by the installation script.

### AMSUA\_CLCOEFS.DAT

Sequential file in ASCII text

Self-documented (lines of comments begin with "#").

Contains the values of the AMSU-A secondary coefficients used in calibration.

There is one file for all the satellites with different sections for :

- AMSU-A1 of NOAA15      ## AMSU-A1 FM1 DATA ## ## ID of instrument --> 9
- AMSU-A2 of NOAA15      ## AMSU-A2 PFM DATA ## ## ID of instrument --> 6
- AMSU-A1 of NOAA16      ## AMSU-A1 PFM DATA ## ## ID of instrument --> 5
- AMSU-A2 of NOAA16      ## AMSU-A2 FM1 DATA ## ## ID of instrument --> 10

This file must be modified in the following cases :

- Insertion of the parameters of a new satellite (furnished just before the satellite launch).

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 13 (see **amsuacl.ksh**)

Located in the directory ../AAPP/src/calibration/libamsuacl and copied into the directory \${PAR\_CALIBRATION\_COEF}/amsua by the installation script.

### Outputs :

#### AMSU-A LEVEL 1B DATA FILE :

Named **aman.11b**

File is renamed at the end of **AAPP\_RUN**

**amsual1b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.11b**

Compared to level.1a structure, 'calibration' parameters have been updated.

Associated with logical unit 11 (see **amsuacl.ksh**).

Located in the directory \${WRK}.

More details, see outputs of **decommutation**.

#### MONAMSUA.TXT

Formatted file in ASCII text.

Filled during the run of **amsuacl.exe** if specified in input options. One record added for one run.

With AAPP version 3, nothing is written into this file.

Associated with logical unit 14 (see **amsubcl.ksh**).

Located in the directory \${PAR\_CALIBRATION\_MONITOR}/noaaXX with XX satellite number.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### SUMMARY FILE FOR PASS :

Sequential file in ASCII text.

Named **amsuacl.log**

The commands “print\*”, “write(\*, )” and the calls to subroutines ml\_wt.. write into it.

Located in the directory \${WRK}.

#### 4.2.16. Inputs/outputs AMSU-B calibration (AMSUBCL)

##### *Inputs :*

---

##### AMSU-B LEVEL 1A DATA FILE :

Named **ambn.l1b**

Output of the decommutation task.

Associated with logical unit 11 (see **amsubcl.ksh**).

Located in the directory \${WRK}.

More details, see outputs of **decommutation**.

##### AMSUB\_CLPARAMS.DAT

Sequential file in ASCII text.

Self-documented (lines of comments begin with "#").

Used for AMSU-B calibration.

There is one file for all the satellites with different sections for :

- AMSU-B of NOAA15      ## AMSU-B PFM DATA ## ## ID of instrument → 4
- AMSU-B of NOAA16      ## AMSU-B FM2 DATA ## ## ID of instrument → 8
- AMSU-B of NOAA17      ## AMSU-B FM3 DATA ## ## ID of instrument → 12
- Values for Fundamental Constants are common for all the satellites.

This file must be modified in the following cases :

- Insertion of the parameters of a new satellite (furnished just before the satellite launch).

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 12 (see **amsubcl.ksh**).

Located in the directory ../AAPP/src/calibration/libamsubcl and copied into the directory \${PAR\_CALIBRATION\_COEF}/amsub by the installation script.

##### AMSUB\_CLCOEFS.DAT

Sequential file in ASCII text.

Self-documented (lines of comments begin with "#").

Contains the values of the AMSU-B secondary coefficients used in calibration.

There is one file for all the satellites with different sections for :

- AMSU-B of NOAA15      ## AMSU-B PFM DATA ## ## ID of instrument → 4
- AMSU-B of NOAA16      ## AMSU-B FM2 DATA ## ## ID of instrument → 8
- AMSU-B of NOAA17      ## AMSU-B FM3 DATA ## ## ID of instrument → 12

This file must be modified in the following cases :

- Insertion of the parameters of a new satellite (furnished just before the satellite launching).

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 13 (see **amsubcl.ksh**).

Located in the directory ../AAPP/src/calibration/libamsubcl and copied into the directory \${PAR\_CALIBRATION\_COEF}/amsub by the installation script.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

### **AMSUB\_BIAS.DAT**

Sequential file in ASCII text

Self-documented (lines of comments begin with ";").

Contains the values of the AMSU-B bias correction for NOAA15

Associated with logical unit 17 (see **amsubcl.ksh**).

Located in the directory ../AAPP/src/calibration/libamsubcl and copied into the directory \${PAR\_CALIBRATION\_COEF}/amsub by the installation script.

#### *Outputs :*

---

### **AMSU-B LEVEL 1B DATA FILE :**

Named **ambn.11b**

File is renamed at the end of **AAPP\_RUN**

**amsubl1b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.11b**

Compared to level.1a structure, 'calibration' parameters have been updated.

Associated with logical unit 11 (see **amsubcl.ksh**).

Located in the directory \${WRK}.

More details, see outputs of **decommutation**.

### **MONAMSUB.TXT**

Formatted file in ASCII text.

Filled during the run of **amsubcl.exe** if specified in input options. One record added for one run.

With AAPP version 3, nothing is written into this file.

Associated with logical unit 14 (see **amsubcl.ksh**)

Located in the directory \${PAR\_CALIBRATION\_MONITOR}/noaaXX with XX satellite number.

### **SUMMARY FILE FOR PASS :**

Sequential file in ASCII text.

Named **amsubcl.log**

The commands "print\*", "write(\*, )" and the calls to subroutines ml\_wt.. write into it.

Located in the directory \${WRK}.

## **4.2.17. Inputs/outputs MHS calibration (MHSCl)**

#### *Inputs :*

---

### **MHS LEVEL 1A DATA FILE :**

Named **ambn.11b**

Output of the decommutation task.

Associated with logical unit 11 (see **mhsc1.ksh**).

Located in the directory \${WRK}.

More details, see outputs of **decommutation**.

### **MHS\_CLPARAMS.DAT**

Sequential file in ASCII text.

Self-documented (lines of comments begin with "#").

Used for MHS calibration.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

There is one file for all the satellites with different sections for :

- MHS of NOAA-N ## MHS PFM DATA on NOAA-18 ## ## ID of instrument → 1
- MHS of METOP-A and METOP simulator
- MHS for NOAA-N' and other METOP satellites will be added at a later date
- Values for Fundamental Constants are common for all the satellites.

This file must be modified in the following cases :

- Insertion of the parameters of a new satellite (furnished just before the satellite launch).

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 12 (see **mhsc1.ksh**).

Located in the directory ../AAPP/src/calibration/libmhsc1 and copied into the directory \${PAR\_CALIBRATION\_COEF}/mhs by the installation script.

### **MHS\_CLCOEFS.DAT**

Sequential file in ASCII text.

Self-documented (lines of comments begin with "#").

Contains the values of the AMSU-B secondary coefficients used in calibration.

There is one file for all the satellites with different sections for :

- MHS of NOAA-N ## MHS PFM DATA ## ## ID of instrument → 1
- MHS of METOP-A and METOP simulator
- MHS for NOAA-N' and other METOP satellites will be added at a later date

This file must be modified in the following cases :

- Insertion of the parameters of a new satellite (furnished just before the satellite launching).

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 13 (see **mhsc1.ksh**).

Located in the directory ../AAPP/src/calibration/libmhsc1 and copied into the directory \${PAR\_CALIBRATION\_COEF}/mhs by the installation script.

### ***Outputs :***

---

#### **MHS LEVEL 1B DATA FILE :**

Named **ambn.11b**

File is renamed at the end of **AAPP\_RUN**

**amsubl1b\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.11b**

Compared to level.1a structure, 'calibration' parameters have been updated.

Associated with logical unit 11 (see **amsubl.ksh**).

Located in the directory \${WRK}.

More details, see outputs of **decommutation**.

#### **MONAMSUB.TXT**

Formatted file in ASCII text.

Filled during the run of **amsubl.exe** if specified in input options. One record added for one run.

With AAPP version 3, nothing is written into this file.

Associated with logical unit 14 (see **amsubl.ksh**)

Located in the directory \${PAR\_CALIBRATION\_MONITOR}/noaaXX with XX satellite number.

#### **SUMMARY FILE FOR PASS :**

Sequential file in ASCII text.

Named **amsubl.log**

The commands "print\*", "write(\*, )" and the calls to subroutines ml\_wt... write into it.

Located in the directory \${WRK}.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.2.18. Inputs/outputs AVHRR calibration (AVHRCL)

##### Inputs :

---

##### **AVHRR LEVEL 1A DATA FILE :**

Named **hrpt.11b**.

Output of the decommutation task.

Associated with logical unit 10 (see **avhrcl.ksh**).

Located in the directory `${WRK}`.

More details, see outputs of **decommutation**.

##### **AVHCAL.TXT**

Sequential file in ASCII text.

Self-documented (lines of comments begin with "#").

Contains calibration parameters.

One file for all the satellites (with 1 section for each).

C1 and C2 constants, used in the function of Planck are the same for all the satellites.

This file must be modified in the following cases :

- Insertion of the parameters of a new satellite (furnished just before the satellite launch).

The version number and the date of the file allow to distinguish the successive versions.

Associated with logical unit 11 (see **avhrcl.ksh**).

Located in the directory `../AAPP/src/calibration/libavhrcl` and copied into the directory `${PAR_CALIBRATION_COEF}/avhcl` by the installation script.

##### Outputs :

---

##### **AVHRR LEVEL 1B DATA FILE :**

Named **hrpt.11b**

File is renamed at the end of **AAPP\_RUN**

**hrpt\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.11b**

Compared to level.1a structure, 'calibration' parameters have been updated.

Associated with logical unit 10 (see **avhrcl.ksh**)

Located in the directory `${WRK}`.

More details, see outputs of **decommutation**.

##### **MONAVHR.TXT**

Formatted file in ASCII text

Contains various statistics parameters showing the evolution of the calibration coefficient calculation.

Filled during the run of **avhrcl.exe** if specified in input options. One record added for one run.

Associated with logical unit 12 (see **avhrcl.ksh**).

Located in the directory `${PAR_CALIBRATION_MONITOR}/noaaXX` with XX satellite number.

##### **SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Named **avhrcl.log**.

The commands "print\*", "write(\*, )" and the calls to subroutines `ml_wt..` write into it.

Located in the directory `${WRK}`.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.2.19. Inputs/outputs sounders calibration application (ATOVIN)

##### Inputs :

---

##### **LEVEL 1B DATA FILES :**

Direct access and unformatted binary files separated for each instrument according to the input options.(one file per instrument).

These files come from HRPT raw data processed by the decommutation, navigation and calibration modules (output files of **hirscl**, **msucl**, **amsuac1**, **amsubcl**, **mhscl**).

Files are named :

**hrsn.11b msun.11b aman.11b ambn.11b**

From AAPP v7.2, the user may specify different input file names, via the “-f ” option.

Each file contains: 1 header record +1 data record for each scan line

The size of the record depends on the instrument:

- 4608 bytes for HIRS
- 2560 bytes for AMSU-A
- 3072 bytes for AMSU-B/MHS
- 1024 bytes for MSU

Each record contains calibration coefficients + counts + time + lat /lon + view angles, altitude and attitude + quality control information + housekeeping information.

For the HIRS, AMSU-A, AMSU-B and MHS the level 1b files are very close to the NOAA 1b formats. The differences are in some scaling factors.

For the MSU, AAPP has developed its own MSU.11b format. It is very close to the HIRS, AMSU-A and AMSU-B formats.

For all the instruments, there are no missing lines (different from NOAA format)

To get the details of the files, see the corresponding include files.

Associated with logical units (see **atovin.ksh**)

- 11 for **hrsn.11b**
- 12 for **aman.11b**
- 13 for **ambn.11b**
- 14 for **msun.11b**

Located in the directory `${WRK}`

##### **FIXED DATA FILE :**

Sequential file in ASCII text.

Named **fdf.dat** containing fixed data for **ATOVIN**.

One file for all the satellites (with 1 section for each).

Self-documented (lines of comments begin with "!").

Contains Satellite name, NOAA satid, nominal satellite height (km), orbit period (sec)

If ATOVS satellite, contains antenna efficiencies for Earth-, platform-, space-view (Ae, Ap, As). For details of the antenna efficiencies see [5].

Note that comment lines must not appear between the 'channel number' and the efficiencies, for each channel.

Optionally contains antenna reflectivity factors for use in the scan-dependent correction – primarily for MHS. See Scientific Description.

ATOVIN will not read beyond a line with 'END' as the first 3 characters.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

This file must be modified in the following case:

- Insertion of the parameters of a new satellite (furnished just before the satellite launch).

Associated with logical unit 10 (see **atovin.ksh**).

Located in the directory `../AAPP/src/preproc/libatovin` and copied into the directory `${DIR_PREPROC}` by the installation script.

#### **STX1\_MAR99CORR.DAT :**

Sequential file in ASCII text.

Contains March 99 STX-1 corrections for NOAA-15 AMSU-B data.

To get details of the format, see the module **amb\_getstx1.F** (`AAPP/src/preproc/libatovin`) that reads the file.

Associated with logical unit 99 (see **atovin.ksh**).

Located in the directory `${DIR_PREPROC}`.

#### **Outputs :**

#### **LEVEL 1C DATA FILES :**

Direct access and unformatted binary files separated for each instrument according to the input options (one file for one instrument).

Named **hrsn.11c msun.11c aman.11c ambn.11c**

From AAPP v7.2, if the user specifies input file names other than the default names, then the output file names will be based on the supplied input files, but with a suffix `.11c` and with "11b" converted to "11c" in the file name.

Files are renamed at the end of **AAPP\_RUN**

**hirs11c\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.11c**

**msul11c\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.11c**

**amsual11c\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.11c**

**amsubl11c\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.11c**

with `SATIMG` : satellite name (example noaa16)

`YYYYMMDD` : year-month-day of data

`HHMN` : hour of data

`NNNN` : orbite number

Each file contains: 1 header record + 1 data record for each scan line.

the record size depends on the instrument:

- 6656 bytes for HIRS
- 3072 bytes for AMSU-A
- 4608 bytes for AMSU-B/MHS
- 512 bytes for MSU

Each record contains brightness temperatures + time + lat/long + view angles, altitude and attitude + quality control info.

Associated with logical units (see **atovin.ksh**):

21 for **hrsn.11c**

22 for **aman.11c**

23 for **ambn.11c**

24 for **msu.11c**

Located in the directory `${WRK}`.

To get the details of the files, see the corresponding include files.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### SUMMARY FILE FOR PASS :

Sequential file in ASCII text.

Named **atovin.log**.

The commands “print\*”, “write(\*, )” and the calls to subroutines ml\_wt.. write into it.

Located in the directory \${WRK}.

#### 4.2.20. Inputs/outputs sounders mapping(ATOVPP)

##### Inputs :

---

#### LEVEL 1C DATA FILES :

Named **hrsn.l1c msun.l1c aman.l1c ambn.l1c iasi.l1c**

Outputs of the **atovin** task.

Associated with logical units (see **atovpp.ksh**):

- 11 for **hrsn.l1c**
- 12 for **aman.l1c**
- 13 for **ambn.l1c**
- 14 for **msu.l1c**
- 15 for **iasi.l1c**
- 16 for **iasi.lpc**
- 17 for **atms.l1c**
- 18 for **cris.l1c**

Located in the directory \${WRK}

More details, see outputs of atovin.

From AAPP v7.2, the user may specify different input file names, via the “-f” option.

#### INSTRUMENT FIXED DATA FILES :

Sequential file in ASCII text.

One file for each instrument, named **HIRS.fdf**, **MSU.fdf**, **AMSUA.fdf**, **AMSUB.fdf**, **IASI.fdf** and containing fixed data for **ATOVPP**.

Data do not depend on the satellite.

Self-documented (lines of comments begin with "!").

Sections are identified by key words starting in column one: **BIAS**, **PREPRO**, **MSULIMB** . Lines before the start of sections are ignored. Some sections are optional in that if they are omitted, ATOVPP will use default values. Sections can be specified in any order.

Data in section **BIAS** are added to the brightness temperatures before any other processing occurs.

Data in section **PREPRO** are the coefficients, thresholds, and other numbers required for the various pre-processing tests and corrections.

Data in section **MSULIMB**, only in the MSU fixed data file, represents the expected differences (in K) between MSU brightness temperatures at each HIRS fov and at nadir. There are two curves, one appropriate for land and one for sea. The intention is to aid the mapping of MSU to HIRS.

ATOVPP will not read beyond a line with 'END' as the first 3 characters.

Associated with logical units (see **atovpp.ksh**):

- 41 for **HIRS.fdf**
- 42 for **AMSUA.fdf**
- 43 for **AMSUB.fdf**
- 44 for **MSU.fdf**
- 45 for **IASI.fdf**



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

54 for **ATMS.fdf**

55 for **CRIS.fdf**

Other data files for IASI:

46 for IASI\_eig\_encode.dat (eigenvectors to be used for Principal Components encoding)

47 for IASI\_eig\_decode.dat (eigenvectors to be used for decoding – not used in atovpp but generated automatically for down-stream use)

48 for IASI\_noise.dat (IASI noise profile for all 8461 channels)

49 for DeApod\_ratio.txt (data to allow de-apodisation of IASI data)

Located in the directory ../AAPP/src/preproc/libatovpp and copied into the directory

\${DIR\_PREPROC} by the installation script.

### LUT FIXED DATA FILE

Sequential file in ASCII text

Named **LUT.fdf**, containing time/angle corrections for the mappings between instrument grids.

Data can depend on the satellite.

Self-documented (lines of comments begin with "!").

Time and angle corrections can be specified for any and all possible mappings. If corrections for a mapping are not specified in the file, then ATOVPP sets them to zero.

ATOVPP will read the file until it comes to a line that isn't a comment line ('!' in column one). It will interpret the line by looking for the satellite name (e.g. 14 for NOAA-14), and also picking out the first two instrument names that it recognises. The first is taken as the mapping instrument, and the second as the target. It then reads the corrections.

ATOVPP will not read beyond a line with 'END' as the first 3 characters.

Associated with logical unit 50 (see **atovpp.ksh**).

Located in the directory ../AAPP/src/preproc/libatovpp and copied into the directory

\${DIR\_PREPROC} by the installation script.

### TOPOGRAPHY FILES

Binary file

Named **maptopog.dat** and **mapbitls.dat**.

Derived from those provided with the CIMSS ITPP export package.

Are two complementary files: a land/sea bitmap and a dataset of surface elevations.

Data are given on a regular 1/6th degree x 1/6th degree lat/lon grid. The surface elevations are to the nearest 100 feet (=30.5metres) and are only specified for land points. This gives a considerable space saving but leads to inaccuracies in some areas (e.g. Lake Victoria).

(Subroutine **surfelev** gives some information).

Associated with logical units (see **atovpp.ksh**): 51 for **mapbitls.dat**

52 for **maptopog.dat**

Located in the directory ../AAPP/src/preproc/libatovpp and copied into the directory

\${DIR\_PREPROC} by the installation script.

### PPBG2A.DAT

Sequential file in ASCII text.

Associated with logical unit 70 (see **atovpp.ksh**).

Located in the directory ../AAPP/src/preproc/libatovpp and copied into the directory

\${DIR\_PREPROC} by the installation script.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

***Outputs :***

---

**LEVEL 1D DATA FILE :**

Direct access and unformatted binary files separated for each target instrument according to the input options (one file for each target instrument, for each instrument grid).

Instrument combinations

- HIRS + AMSU-A + AMSU-B/MHS data on the HIRS grid,
- AMSU-A + AMSU-B/MHS data on the AMSU-A grid,
- AMSU-A + AMSU-B/MHS data on the AMSU-B grid,
- AMSU-A + MHS data on the IASI grid,
- HIRS + MSU data on the HIRS grid.

In the standard AAAPP\_RUN\_NOAA script there is only one target instrument: HIRS. So only one output level 1D file: **hirs.11d**. The User can modify the call to **atovpp** if other combinations are required.

From AAPP v7.2, if the user specifies input file names other than the default names, then the output file names will be based on the supplied input files, but with a suffix .11d and with "11c" converted to "11d" in the file name.

File **hirs.11d** is renamed at the end of **AAPP\_RUN**

**hirs11d\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.11d**

with SATIMG : satellite name (example noaa16)

YYYYMMDD.: year-month-day of data

HHMN : hour of data

NNNN : orbit number

Each file contains: 1 header record + 1 data record for each scan line  
the record size depends on the instrument:

15872 bytes for HIRS

Each record contains pre-processed brightness temperatures + time + lat/long + satellite zenith angle + azimuth angle + altitude and attitude + quality control information + pre-processing flags + surface information.

Associated with logical unit 21 (see **atovpp.ksh**).

Located in the directory **\${WRK}**

To get the details of the file, see the corresponding include file.

Note: **atovpp** pre-processes brightness temperatures on grid of selected instruments: HIRS, AMSU-A, AMSU-B, IASI. This format of output on each grid (HIRS, AMSU-A, AMSU-B, IASI) is intended to be flexible. Some parts of the format are fixed, and other parts will be customised to fit the requirements of individual centers. The AMSU-A and B level 1d formats may need to be expanded to accommodate extra mappings. A change in format will require changes in the *ppXouth* and *ppXoutd* routines.

**SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Named **atovpp.log**

The commands "print\*", "write(\*, )" and the calls to subroutines **ml\_wt..** write into it.

Located in the directory **\${WRK}**.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.2.21. Inputs/outputs for mapping cloud mask AVHRR to HIRS (AVH2HIRS)

##### *Inputs :*

---

##### **USER INPUT PARAMETERS IN ATOVS\_ENV :**

Set up in the following run conditions. There are two possibilities to run avh2hirs: with MAIA and without MAIA. With MAIA, the cloud detection is done and cloud information is put in the sounder field of view. By default this script is used without MAIA. To run MAIA, set an environment variable RUN\_MAIA=yes. MAIA needs forecast files and ancillary data, see section 4.4 on MAIA to set the environment for MAIA.

##### **HIRS LEVEL 1D DATA FILE :**

Named **hirs.l1d**.

Outputs of the **atovpp** pre-processing task of mapping AMSU-A/AMSU-B or MSU into a HIRS grid. Associated with logical unit 12 (see **avh2hirs.ksh**).

Located in the directory `${WRK}`.

More details, see outputs of **atovpp**.

##### **AVHRR LEVEL 1B DATA FILE :**

Direct access and unformatted binary file.

Named **hrpt.l1b**.

Output of **avhrc1** AVHRR calibration and localisation task.

File contains: 1 header record + 1 data record for each scan line.

The size of the record: 22016 bytes (does not respect 1B NOAA size, see appendix A).

No missing line (different from NOAA format).

Each data record contains counts + time + calibration coefficients + lat/long + housekeeping information + quality control information.

Associated with logical unit 11 (see **avh2hirs.ksh**).

Located in the directory `${WRK}`.

To get the details of the files, see the corresponding include files.

##### **TIME AND ANGLE CORRECTION FILE :**

Sequential file in ASCII text, including time and angle corrections for mapping.

Named **cor\_nxx.dat**, xx satellite number (cor\_n12.dat, cor\_n14.dat, cor\_n15.dat, cor\_n16.dat).

Contains optional corrections and adjustments for mapping (used by **lutmap**). Zeros are used by default.

To get the details of the files, see modules **avh2hirs** or **avh2hirs\_atovs** that read the file.

Associated with a constant logical unit lucor=50+xx, xx satellite number.

Located in the directory `../AAPP/src/preproc/libavh2hirs` and copied into the directory `${DIR_PREPROC}` by the installation script.

##### *Outputs :*

---

##### **HIRS LEVEL 1D DATA FILE WITH CLOUD MASK :**

Named **hirs.l1d**

File **hirs.l1d** is renamed at the end of **AAPP\_RUN**

**hirs11d\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNN}.l1d**

with SATIMG : satellite name (example noaa16)

YYYYMMDD.: year-month-day of data

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

HHMN : hour of data

NNNNN : orbite number

Compared to **hirs.11d** input file, the 13 ‘cloud mask’ parameters have been updated for each HIRS target pixel.

Associated with logical unit 12 (see **avh2hirs.ksh**):

Located in the directory `${WRK}`

More details, see outputs of **atovpp**.

#### **STATISTICS FILE :**

Statistics file in formatted ASCII text

Named **mapqual\_\${SATIMG}.txt**.

Filled at the end of **AVH2HIRS** processing.

Contains global H8-A4 standard deviations (F6.5) and H8-A4 standard deviation for each HIRS pixel (56F5.2). Start date (2I3.2) and orbit (I6) are written at the beginning of the file.

Associated with logical unit 22 (see **avh2hirs.ksh**).

Located in the directory `${WRK}`.

#### **SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Named **avh2hirs.log**

The commands “print\*”, “write(\*, )” and the calls to subroutines ml\_wt.. write into it.

Located in the directory `${WRK}`.

### **4.2.22. Inputs/outputs sounders calibration application (AVHRRIN)**

#### ***Inputs :***

---

#### **LEVEL 1B DATA FILES :**

Direct access and unformatted binary files separated for AVHRR instrument

The file comes from HRPT raw data processed by the decommutation, navigation and calibration modules

Named **hrpt.11b**

File is renamed at the end of **AAPP\_RUN**

`hrpt_${SATIMG}_${YYYYMMDD}_${HHMN}_${NNNNN}.11b`

Located in the directory `${WRK}`.

The file contains: 1 header record +1 data record for each scan line

The size of the record depends on the instrument:

- 22016 bytes

Each record contains calibration coefficients + counts + time + lat /lon + view angles, altitude and attitude + quality control information + housekeeping information.

There are no missing lines (different from NOAA format)

To get the details of the files, see the corresponding include files.

Associated with logical units AVH1Bunit (see **ATOVS\_ENV7**)

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

***Outputs :***

---

**LEVEL 1C DATA FILES :**

Direct access and unformatted binary files separated for each instrument according to the input options (one file for one instrument).

Named **avhrr.11c** by default

File is renamed at the end of **MAIA3\_RUN**

**avh\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.11c**

with SATIMG : satellite name (example noaa16)

YYYYMMDD.: year-month-day of data

HHMN : hour of data

NNNNN : orbite number

The file contains: 1 header record + 1 data record for each scan line.

the record size: 29808 bytes

Each record contains brightness temperatures + time + lat/long + view angles, altitude and attitude + quality control info.

Associated with logical units AVH1Cunit (see **ATOVS\_ENV7**)

Located in the directory **\_\${WRK}**.

To get the details of the files, see the corresponding include files.

**SUMMARY FILE FOR PASS :**

Sequential file in ASCII text.

Named **avhrrin.log**.

The commands "print\*", "write(\*, )" and the calls to subroutines ml\_wt.. write into it.

Located in the directory **\_\${WRK}**.

File is renamed at the end of **MAIA3\_RUN**

**avhrrin\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.log**

**4.2.23. Inputs/outputs for conversion AVHRR AAPP I1b format to AVHRR PFSL1B format (aapp-eps\_avhrr11b)**

***Inputs :***

---

**AVHRR LEVEL 1B DATA FILE :**

Named **hrpt.11b**

File is renamed at the end of **AAPP\_RUN**

**hrpt\_\${SATIMG}\_\${YYYYMMDD}\_\${HHMN}\_\${NNNNN}.11b**

Compared to level.1a structure, 'calibration' parameters have been updated.

Associated with logical unit 10 (see **avhrcl.ksh**)

Located in the directory **\_\${WRK}**.

More details, see outputs of **decommutation**.

***Outputs :***

---

**AVHRR PFS LEVEL 1B DATA FILE :**

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.2.24. Inputs/outputs for SATEPH navigation tool.

**sateph** calls modules: satpos.xxx.exe and ephe. For the different files, the origin of inputs and outputs have been specified.

##### *Inputs :*

---

#### ***TBUS\_YYYYMMDD.TXT OR TLE\_YYYYMMDD\_HHMN.TXT OR SPM\_\*.TXT***

- Input for satpost.exe satpostle.exe satposspm.exe
- See inputs/outputs for satpost, satpostle, satposspm

#### ***TBUS\_SSSS.INDEX OR TLE\_SSSS.INDEX OR SPM\_SSSS.INDEX***

- Input for satpost.exe satpostle.exe satposspm.exe
- See inputs/outputs for satpost, satpostle, satposspm

##### *Outputs :*

---

#### ***SATPOS\_SSSS\_YYYYMMDD.TXT***

- output for satpost.exe or satpostle.exe or satposspm.exe
- input for ephe
- See inputs/outputs for satpost, satpostle, satposspm

#### ***EPHE\_SSSS\_YYYYMMDD.TXT***

- Output of ephe
- Name of the ASCII ephemeris file associated with a given station and a specific satellite, xx satellite number, yyyyymmdd start date of the ephemeris.
- Located in the directory \${DIR\_NAVIGATION}/ephe.
- Each data line contains the following information : calendar date of the event (yyyy/mm/dd), time of the event (hh:mm:ss.sss), satellite name (noaaxx), orbit number, event code (start\_acq : start of acquisition, stop\_acq: end of acquisition, asc\_node : ascending node, dsc\_node: descending node, sun\_rise : sun rise for station, sun\_set: sun set for station), a text associated with the event (station name for start\_acq/stop\_acq, longitude of nodes (deg) for asc\_node/dsc\_node). No line of comments authorised.
- More details are given in *ephe.5*.

#### 4.2.25. Inputs/outputs for LGEPHEING navigation tool

##### *Inputs :*

---

#### ***TBUS\_YYYYMMDD.TXT***

See above 3.3.2 (inputs/outputs for tbusing)

##### *Inputs/Outputs :*

---

#### ***LGEPHE\_NOAAXX.INDEX***

Name of the ASCII long-term ephemeris file associated with a given station and a specific satellite, xx satellite number.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

Located in the directory  $\{\text{DIR\_NAVIGATION}\}/\text{lgephe}$ .

Contains all the needed orbital parameters for long ephemeris calculation.

- The first line contains the NOAA name of the satellite.
- Each data line contains the following information : epoch time of ascending node in CNES Julian day (day 0 = 01/01/50 0h), string for epoch time (yyyy/mm/dd hh:mm:ss.sss), orbit number, longitude of the ascending node (deg), longitude increment (deg), semi-major axis (km), inclination (deg), and nodal period (hh:mm:ss.sss).

More details are given in *lgephe.5*.

***Outputs :***

---

**SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Named **lgepheing.log**

The commands “print\*”, “write(\*, )” and the calls to subroutines ml\_wt.. write into it.

**4.2.26. Inputs/outputs for LGEPHE navigation tool**

***Inputs :***

---

**STATIONS.TXT**

Name of the ASCII file containing geographic coordinates of reception station

Located in the directory  $\{\text{DIR\_STATIONS}\}$ .

Each line contains following informations : latitude(deg)/longitude(deg)/altitude(km), elevation min. (deg), and name.

**LGEPHE\_NOAAXX.INDEX**

See 3.3.17 Inputs/Outputs for lgepheing navigation tools.

***Outputs :***

---

**LGEPHE\_NOAAXX\_YYYYMMDD.TXT**

Name of the ASCII long-term ephemeris file associated with a list of stations and a specific satellite, xx satellite number, yyyyymmdd ephemeris start date.

Located in the directory  $\{\text{DIR\_NAVIGATION}\}/\text{lgephe}$ .

Each data line contains the following information : calendar date of the event (yyyy/mm/dd), time of the event (hh:mm:ss.sss), satellite name (noaaxx), orbit number, event code (start\_acq : start of acquisition, stop\_acq: end of acquisition, asc\_node : ascending node, dsc\_node: descending node), a text associated with the event (station name for start\_acq/stop\_acq, longitude of nodes (deg) for asc\_node/dsc\_node).

No line of comments authorised.

**SUMMARY FILE FOR PASS :**

Sequential file in ASCII text .

Named **lgephe.log**

The commands “print\*”, “write(\*, )” and the calls to subroutines ml\_wt.. write into it.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.2.27. Inputs/outputs for ALLEPH navigation tool

**alleph** calls several modules: satpost.exe, ephe, tracking, antcnft. For the different files, the origin of inputs and outputs have been specified.

##### *Inputs :*

---

#### **TBUS\_YYYYMMDD.TXT**

Input for satpost.exe  
See above 3.3.2 (inputs/outputs for tusing)

#### **TBUS\_NOAAXX.INDEX**

Input for satpost.exe  
See above 3.3.2 (inputs/outputs for tusing)

**Or**

#### **TLE\_YYYYMMDD\_HHMN.TXT**

Input for satpostle.exe  
See above 3.3.3 (inputs/outputs for tleing)

#### **TLE\_NOAAXX.INDEX**

Input for satpostle.exe  
See above 3.3.3 (inputs/outputs for tleing)

##### *Outputs :*

---

#### **SATPOS\_NOAAXX\_YYYYMMDD.TXT**

output for satpost.exe or satpostle.exe  
input for ephe and tracking  
See above 3.3.4 and 3.3.5 (inputs/outputs for satpost/satpostle)

#### **EPHE\_NOAAXX\_YYYYMMDD.TXT**

Output of ephe and tracking  
Name of the ASCII ephemeris file associated with a given station and a specific satellite, xx satellite number, yyyyymmdd start date of the ephemeris.  
Located in the directory \${DIR\_NAVIGATION}/ephe.  
Each data line contains the following information : calendar date of the event (yyyy/mm/dd), time of the event (hh:mm:ss.sss), satellite name (noaaxx), orbit number, event code (start\_acq : start of acquisition, stop\_acq: end of acquisition, asc\_node : ascending node, dsc\_node: descending node, sun\_rise : sun rise for station, sun\_set: sun set for station, site\_max: maximum site during the acquisition, short\_acq: acquisition too short, start\_conflict or stop\_conflict: start/stop of conflict for one antenna acquisition system), a text associated with the event (station name for start\_acq/stop\_acq/start\_conflict/stop\_conflict, longitude of nodes (deg) for asc\_node/dsc\_node, site (deg) for site\_max, acquisition duration (in minutes) for short\_acq). No line of comments authorised. More details are given in *ephe.5*.

#### **EPHE\_YYYYMMDD.TXT**

Input/output of antcnft



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

Name of the ASCII ephemeris file associated with a given station and several satellites, yyyyymmdd ephemeris start time.

Located in the directory \${DIR\_NAVIGATION}/ephe.

Same data lines as above.

#### **TRACKING\_NOAAXX\_YYYYMMDD\_OOOOO.TXT**

Name of the ASCII tracking angle file associated with a SATPOS file, xx satellite number, yyyyymmdd date at start of acquisition, ooooo is the orbit number at start of acquisition.

Located in the directory \${DIR\_NAVIGATION}/tracking.

The header contains the NOAA name of the satellite, ground station latitude and longitude (deg), ground station altitude (km) and minimum site (deg), processing time and tracking start time (dd/mm/yy hh:mm:ss.sss), orbit number at start time, the time step value (in seconds), and text describing data lines.

Each data line contains the following information: site (deg), azimuth counted anticlockwise with origin at south direction (deg), corresponding date (dd/mm/yy hh:mm:ss.sss).

More details are given in *tracking.5*.

#### **4.2.28. Inputs/outputs for TBUSDISP navigation tool**

**tbusdisp** is an interactive script that displays the content of a TBUS message

##### *Inputs :*

---

#### ***TBUS\_SSSS.INDEX***

- index file for considered satellite *ssss*

#### ***INTERACTIVE COMMANDS***

- satellite name
- date

##### *Outputs :*

---

#### ***STANDARD OUTPUT***

- *TBUS* displayed on standard output

#### **4.2.29. Inputs/outputs for TLEPRINT navigation tool**

**tbusprint** is an interactive script that displays the content of a 2-Line message

##### *Inputs :*

---

#### ***INTERACTIVE COMMANDS***

- file name of a 2 line file
- satellite name

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

*Outputs :*

---

## **STANDARD OUTPUT**

- 2-Line displayed on standard output

### **4.2.30. Inputs/outputs for EPHE, TRACKING, ANTCNFT, DRIFTEPHE navigation tool**

All those scripts are dummy scripts in order to interface shell with fortran. See the relative commands ephe.exe tracking.exe antcnft.exe and driftephe.exe

## **4.3. DYNAMIC ARTICULATION**

In this paragraph the text basic information parameters are :

- s : satellite (e.g. : noaa14)
- d : date (yyyymmdd) (e.g. :19980512)
- h : hour-minute (hhmm) (e.g. :1415)
- n : orbit number (nnnn) (e.g. :1750)

### **4.3.1. Description of the main script AAPP\_RUN\_NOAA**

With the **AAPP\_RUN\_NOAA** korn shell, all the different steps hang together: From a HRPT data file, HIRS.11b, MSU.11b, AMSU-A.11b, AMSU-B.11b, MHS.11b, HIRS.11c, MSU.11c, AMSU-A.11c, AMSU-B/MHS.11c, HIRS.11d are created. It tests the satellite number to identify the type of data, TOVS or ATOVS data.

Usage is

```
AAPP_RUN_NOAA [-D] [-Y year] [-i instruments ] [-g grids ] [-o outdir ] [-z] [-C] [-L] file_name
  -D          : debug on
  -Y year     : year of the HRPT data (default=current year)
  -i instruments : from the list "AMSU-A AMSU-B HIRS MSU AVHRR DCS"
                (default all available)"
  -g grids    : from the list "AMSU-A AMSU-B HIRS" (default "HIRS")
  -z          : skip avh2hirs
  -C          : skip calibration
  -L          : skip Earth location
  file_name   : HRPT data file (full path or relative to current)
```

**-D** and **-Y** are optional. But it is strongly recommended to specify the year of the HRPT data. By default, the year is the current year. Using YEAR-default can cause problems when processing later data from current year or earlier.

file\_name is an obligatory parameter.

Calls other scripts :

tbusing, tleing, satpost, satpostle, decommutation, prhirs, hirscl, hirscl\_algoV4, prhmsu, msucl, amsuacl, amsubcl, mhsccl, prhavh, avhrccl, atovin, atovpp, avh2hirs, log\_info, log\_error

Calls executable files :

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

hrpidf.exe, sdh2orbnnum.exe

#### 4.3.2. Description of the script CHK1BTIME

Included in the decommutation.ksh file

The script **chk1btime** is activated with one obligatory argument: The level 1 b file name.

For example:       chk1btime hirs11a\_noaa15\_19980716\_0715\_00905.11b

#### 4.3.3. Description of the script TBUSING

(See also the reference manual man pages *tbusing.1*)

With the **tbusing.ksh** korn shell and after each performance of **tbusing.exe**, historical files (automatically determined by input satellites numbers) are updated.

Usage is:

**tbusing** [-s satellite] [-f tbus\_file]

-s to specify the list of satellites to be considered

-f to specify the TBUS bulletin to process.

-s and -f are optional.

If no parameter is specified as an option, defaults are:

-s noaa09 noaa11 noaa12 noaa14

(see the variable PAR\_NAVIGATION\_DEFAULT\_LISTESAT\_INGEST\_TBUS in the script)

-f : all the TBUS bulletins which are newer than the last update of the index files corresponding to the satellite list.

#### 4.3.4. Description of the script GET\_TLE

**get\_tle** to retrieve current 2-Line orbital elements from a web site

The usage is : **get\_tle**

all parameters are loaded from the configuration file

#### 4.3.5. Description of the script TLEING

(See also the reference manual man pages *tleing.1*)

With the **tleing.ksh** korn shell and after each performance of **tleing.exe**, historical files (automatically determined by input satellites numbers) are updated.

Usage is:

**tleing** [-s satellite] [-f tle\_file] [-c]

-s to specify the list of satellites to be considered

-f to specify the TLE bulletin to process.

-c to check presence of input 2lines files in final index file

-c -s and -f are optional.

If no parameter is specified as an option, defaults are:

-s : value of the variable PAR\_NAVIGATION\_DEFAULT\_LISTESAT\_INGEST\_TLE in the ATOVS\_ENV file)

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**-f** : all the TLE bulletins which are newer than the last update of the index files corresponding to the satellite list.

#### 4.3.6. Description of the script SATPOST

(See also the reference manual man pages *satpost.1*)

For a given satellite and a given acquisition station, the command creates a position-velocity file (SATPOS) using TBUS bulletins.

Usage is:

**satpost** [-o] [-s satellite] [-S station] [-d start date] [-n number of days] [-i increment in seconds] [-c search criteria]

**-o -s -S -d -n -i -c** are optional.

If no parameter is specified as an option, defaults are : noaa14, Lannion, today 0h, 1.0, 120.0, n (n= nearest, p = preceding).

The option **-o** specifies that the data will be stored in the file *satpos\_noaxx\_yyyymmdd.txt*.  
Output default is the standard output..

#### 4.3.7. Description of the script SATPOSTLE

(See also the reference manual man pages *satpostle.1*)

For a given satellite and a given acquisition station, the command creates a position-velocity file (SATPOS) using TLE bulletins.

Usage is:

**satpostle** [-o] [-s satellite] [-S station] [-d start date] [-n number of days] [-i increment in seconds] [-c search criteria]

**-o -s -S -d -n -i -c** are optional.

If no parameter is specified as an option, defaults are : noaa14, Lannion, today 0h, 1.0, 120.0, n (n= nearest, p = preceding).

The option **-o** specifies that the data will be stored in the file *satpos\_noaxx\_yyyymmdd.txt*.  
Output default is the standard output..

#### 4.3.8. Description of the script DECOMMUTATION

**decommutation.ksh** reads the environment parameters in *ATOVS\_ENV7* to get the conditions of the run.

It associates the logical unit number with the needed fixed data *amsua\_clparams.dat*.

It generates dynamically the user input options file *decommutation.inp* and the program is then launched with the user options file as input:

```
decommutation.exe < decommutation.inp
```

The log of the program execution is saved in the output file *decommutation.log*.

At the end, for HIRS, MSU, AMSU-A and AMSU-B, the script calls **chk1btime** script (inside *decommutation.ksh* file) to correct scan line datation for level 1 b files.

**chk1btime** script needs one argument: the complete name of the level 1b file (see also the reference manual man pages *chk1btime.1*).

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

The log file are saved in the output files *decommutation.log*.  
 Lastly, it deletes the input file *decommutation.inp* and the different links.

Usage is:

**decommutation** \${A\_TOVS} decommutation.par \${FILE}

The 3 arguments are obligatory.

\${A\_TOVS} = TOVS for satellite number < or = 14

\${A\_TOVS} = ATOVS for satellite number > 14

In the decommutation.par file, options are written in this order:

\$1,\$2,\$3,\$4,\$5, \$6, \$7, \$8,\$9,\$10,\$11,\$12	!OPTION NUMBERS
\$lu1,\$lu2,\$lu3,\$lu4,\$lu5,\$lu6,\$lu7,\$lu8,\$lu9,\$lu10,\$lu11,\$lu12	!STREAM NO.S
\${YEAR}	! year of the data
0	! operational mode
\${NNNNN},\${NNNNN}	! start and end orbit numbers

with

- \$1 = 0 or 1 for level of error logging
- \$2 = 0 or 1 for HIRS/3 or HIRS/4 (1 indicates extract HIRS/3 or HIRS/4 data)
- \$3 = 0 or 1 for AMSU-A1 (1 indicates extract AMSU-A1 data)
- \$4 = 0 or 1 for AMSU-A2 (1 indicates extract AMSU-A2 data)
- \$5 = 0 or 1 for AMSU-B/MHS (1 indicates extract AMSU-B/MHS data)
- \$6 = 0 or 1 for HIRS/2 (1 indicates extract HIRS/2 data)
- \$7 = 0 or 1 for MSU (1 indicates extract MSU data)
- \$8 = 0 or 1 for DCS (1 indicates extract DCS data)
- \$9 = 0 or 1 for SEM (1 indicates extract SEM data)
- \$10= 0 or 1 for SBUV (1 indicates extract SBUV data)
- \$11= 0 or 1 for SAR (1 indicates extract SAR data)
- \$12= 0 or 1 for AVHRR (1 indicates extract AVHRR data)

- \$lu1 is the logical unit of the log file
- \$lu2 is the logical unit of the HIRS/3 or HIRS/4.11a output file
- \$lu3 is the logical unit of the AMSU-A1.11a output file
- \$lu4 is the logical unit of the AMSU-A2.11a output file
- \$lu5 is the logical unit of the AMSU-B.11a output file
- \$lu6 is the logical unit of the HIRS/2.11a output file
- \$lu7 is the logical unit of the MSU.11a output file
- \$lu8 is the logical unit of the DCS.11a output file
- \$lu9 is the logical unit of the SEM.11a output file
- \$lu10 is the logical unit of the SBUV.11a output file
- \$lu11 is the logical unit of the SAR.11a output file
- \$lu12 is the logical unit of the HRPT.11a output file

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.3.9. Description of the scripts **HIRSCL**, **HIRSCL\_ALGOV4**, **MSUCL**, **AMSUCL**, **AMSUBCL**, **MHSCL**, **AVHRCL**

Those scripts can run alone, outside of the processing.

They read the environment parameters in ATOVS\_ENV8 to get the conditions of the run.

For the navigation of the level 1b file,

They create the SATPOS file if it does not exist by calling the scripts **satpostle** or **satpost**.

They get previous or current orbit attitude values by calling the function **det\_att**.

They define calibration and errorclock related files.

The scripts get the different parameters to generate the input parameters of **hirscl.exe**, **hirscl\_algoV4.exe**, **msucl.exe**, **amsucl.exe**, **amsubcl.exe**, **mhscl.exe**, **avhrcl.exe**.

The level 1b files and the required fixed data files are used without names within the executable.

The names of the files are dynamically built inside the scripts.

The log files are saved in the output files *hirscl.log*, *msucl.log*, *amsucl.log*, *amsub.log*, *mhscl.log*, *avhrcl.log*.

Lastly, all the links between the files and the associated logical units are deleted.

Usage is:

**hirscl [-c] [-l] -s satellite -d yyyyymmdd -h hhmn -n nnnnn source.11b**

**hirscl\_algoV4 [-c] [-l] -s sating -d yyyyymmdd -h hhmm -n nnnnn source.11b**

**msucl [-c] [-l] -s satellite -d yyyyymmdd -h hhmn -n nnnnn source.11b**

**amsucl [-c] [-l] -s satellite -d yyyyymmdd -h hhmn -n nnnnn source.11b**

**amsubcl [-c] [-l] -s satellite -d yyyyymmdd -h hhmn -n nnnnn source.11b**

**mhscl [-c] [-l] -s satellite -d yyyyymmdd -h hhmn -n nnnnn source.11b**

**avhrcl [-c] [-l] -s satellite -d yyyyymmdd -h hhmn -n nnnnn source.11b**

**-c** for calibration.

**-l** for Earth location.

**-s -d -h -n** are the basic information parameters (see above 4.2).

**yyyyymmdd**: year/month/day, **hhmm**: hours/minutes, **nnnnn**: orbit number.

**source.11b** : name of the level 1b file to process

**-c -l** are optional.

**-s -d -h -n** and the **source.11b** are obligatory.

#### 4.3.10. Description of the script **ATOVIN**

This script allows running of the **atovin.exe** program that processes level 1b TOVS/ATOVS to level 1c.

It reads the environment parameters in ATOVS\_ENV7 to get the conditions of the run.

It generates dynamically the user input options file *atovin.input* including the instruments to process. Examples: HIRS MSU or HIRS AMSU-A AMSU-B.

It associates logical unit numbers with level 1b files to read, with level 1c files to write, and with required fixed data.

The program is then launched with the user options file as input.

`atovin.exe < atovin.input`

The log file is saved in the output file *atovin.log*.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

Lastly, it deletes the input file *atovin.input* and the links between the level 1b and level 1c files and the associated logical units.

Usage is:

**atovin [-f infiles] instruments**

If input files are specified, they must be in the same order as the list of instruments, and must be enclosed in quotes if there is more than one instrument.

A companion script **atovin\_antorr** is available to apply or remove the antenna correction for microwave instruments (AMSU/MHS). Usage is:

**atovin\_antcorr [-f infiles] [-z] instruments**

In this case the input files are level 1c. If the *-z* option is supplied, the program will attempt to remove any antenna correction that is already present in the data.

#### 4.3.11. Description of the script ATOVPP

The script allows running of the **atovpp.exe** program that processes level 1c TOVS/ATOVS and IASI to level 1d.

It reads the environment parameters in *ATOVS\_ENV7* to get the conditions of the run.

It associates logical unit numbers with level 1c files to read, with the HIRS level 1d file to write, and with the required fixed data files.

It generates dynamically the user input options file *atovpp.inp* (instruments to read and instrument grids to output depending on whether we have TOVS or ATOVS data).

The program is then launched with the user options file as input.

*atovpp.exe* < *atovpp.inp*.

The log file is saved in the output file *atovpp.log*.

Lastly, it deletes the input file *atovpp.inp* and the links between the level 1c files, the level 1d file and the associated logical units.

Usage is:

**atovpp [-f infiles] [-r] -g grids -i instruments**

where *grids* and *instruments* are sub-sets of "AMSU-A AMSU-B HIRS IASI" in the case of ATOVS, or "MSU HIRS" in the case of TOVS. Quotes are needed if there is more than one grid, instrument or file name. "MHS" can be specified instead of "AMSU-B". If input files are specified, they must be in the same order as the list of instruments.

An alternative syntax for backward compatibility with AAPP version 5 and earlier is

**atovpp [A]TOVS [h]**

in which case if the **h** argument is absent only the HIRS grid is generated. If the **h** argument is present, HIRS and AMSU-B grids are generated.

#### 4.3.12. Description of the script AVH2HIRS

The script reads the parameters file *ATOVS\_ENV* to get the conditions of the run.

It makes links in input with the following files:

- AVHRR level 1b
- HIRS level 1d: From the HIRS level 1d file, it determines the satellite and datation by using the command **l1didf** which opens the file and reads the header.
- Time and angle correction: it also depends on the satellite.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

- Logical unit numbers associated with these files are set up in the script

If the script is run with MAIA, it invokes **maia\_env** which makes links to forecast files, climatologies files, atlas files and threshold files needed by MAIA. See chapter 4.5 MAIA to set the environment for MAIA (environment variables, forecast files and ancillary data). In this case, the AVHRR level 1b file is converted in AVHRR level 1c by **avhrrin** (see below), and then in hdf5 (avh5) by **convert\_to\_hdf5**. Then the script invokes the **avh2hirs\_atovs.exe** or the **avh2\_hirs.exe** command. The log file is saved in the output file *avh2hirs.log*. At the statistics file associated with the logical unit 22 is then saved with the name *mapqual.txt*. Links between logical unit to files are deleted at the end of the script. By default MAIA is not used.

#### 4.3.13. Description of the script AVHRRIN.KSH

This script is invoked as:

```
avhrrin
```

Options can be specified

- i file\_name1 : full pathname of the input hrpt/avhrr 1b file (default \$WRK/hrpt.11b)
- o file\_name2 : full pathname of the output hrpt/avhrr 1c file (default \$WRK/avhrr.11c)

The script reads the parameters file ATOVS\_ENV to get the conditions of the run.

Exit codes: 0 normal end

- 1 bad input parameters, input data, usage,...
- 2 bad output code for avhrrin.exe

#### 4.3.14. Description of the script EPS\_AVHRR1B-MAIN

This script is invoked as follows:

```
aapp-eps_avhrr1b avhrr.11b [ avhrr.pfs ]
```

Where avhrr.11b is a calibrated and navigated avhrr AAPP file. The name of the output PFS file is optional; if it is not passed as an argument, then the program will use the standard PFS filename.

#### 4.3.15. Description of the script EPS\_CONVERT\_IASIL1C

This script is invoked as:

```
convert_iasil1c iasi. pfs_iasil1c
```

where pfs\_iasil1c is the IASI file at level 1c PFS format

#### 4.3.16. Description of the script NOAA\_CLASS\_TO\_AAPP

This script is invoked as:

```
noaa_class_to_aapp inputfile outputfile
```

where inputfile is the NOAA/CLASS file to be converted, in NOAA naming convention, e.g. NSS.HIRX.N[A-P].D?????.\*



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.3.17. Description of the script AVHRR\_AAPP\_TO\_CLASS

This script is invoked as:

```
avhrr_aapp_to_class inputfile outputfile
```

where inputfile is the AAPP AVHRR level 1b file to be converted.

#### 4.3.18. Description of the script SATPOS-SVM

This script is invoked as:

```
satpos-svm satpos.txt [ xxxx_SVM_... ]
```

The name of the SVM file is optional; if it is not present, then stdout is used.

#### 4.3.19. Description of the script MESSAGES-OSV

This script is invoked as:

```
messages-osv messages.txt [ xxxx_OSV_... ]
```

The name of the OSV file is optional; if it is not present, then stdout is used.

#### 4.3.20. Description of the script SATEPH

**sateph** to run the ephemeris scheme (short term)

The usage is : **sateph [-options]**

where options are:

- s** satellite\_list
- S** station\_name
- b** bulletin\_list
- d** start\_date
- n** number of days (real)
- i** increment in seconds (real)
- c** search criteria (n for nearest or p for preceding)

the date format can be a date or a date/hour string or an offset in days to the current day

for example -d 'dd/mm/yy hh:mm:ss.sss' (2 spaces between yy hh)

or -d 'dd/mm/yy hh:mm'

or -d dd/mm/yy

or -d '-3' 3 days ago

or -d '4' 4 days after

#### 4.3.21. Description of the script LGEPHEING

(See also the reference manual man pages *lgepheing.1*)

With the **lgepheing.ksh** korn shell and after each performance of **lgepheing.exe**, historical files (automatically determined by input satellites numbers) are updated. It must run before lgephe.

Usage is:

```
lgepheing [-s satellite_list] [-f tbus_file]
```

**-s** to specify the list of satellites to be considered.

**-f** to specify the TBUS bulletin to process.

**-s** and **-f** are optional.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

If no parameter is specified as an option, defaults are:

**-s** noaa09 noaa11 noaa12 noaa14

(see the variable PAR\_NAVIGATION\_DEFAULT\_LISTESAT\_INGEST\_TBUS in the script)

**-f** : all the TBUS bulletins which are newer than the last update of the index files corresponding to the satellite list.

#### 4.3.22. Description of the script LGEPHE

(See also the reference manual man pages *lgephe.1*)

With the **lgephe.ksh** korn shell and after each performance of **lgephe.exe**, for a given satellite and several given stations, a long-term ephemeris file is created using the ephemeris index file.

Usage is:

**lgephe** [-o] [-s satellite\_name] [-S station\_list] [-d start date] [-n number of days]

**-o -s -S -d -n** are optional.

If no parameter is specified as an option, defaults are: noaa14, Lannion, today 0h, 10.0.

The option **-o** specifies that the data will be stored in the file *lgephe\_noaxx\_yyyymmdd.txt*.

#### 4.3.23. Description of the script ALLEPH

(See also the reference manual man pages *alleph.1*)

With the **alleph.ksh** korn shell performs all the basic commands needed to generate SATPOS files, tracking angle files, ephemeris files etc.. It calls the commands satpos, ephe, tracking and antcnft.

Usage is:

**alleph** [-s satellite] [-S station] [-b bulletin] [-d start date] [-n number of days]

[-i increment in seconds] [-c search criteria] [-o antenna\_steering\_seconds]

**-s -S -b -n -i -c -o** are optional.

If no parameter is specified as option, defaults are:

- For the list of satellites: noaa14 noaa12 noaa11 noaa09
- For the station :Lannion
- For the list of bulletin: tbus tbus tbus tbus
- For the start date: today 0h
- For the number of days: 1.0
- For the increment: 120.0
- For the search criteria: n (n= nearest, p = preceding).
- For the antenna steering duration: 0sec

#### 4.3.24. Description of the command EPHE

(See also the reference manual man pages *ephe.1*)

The command **ephe** creates an ephemeris file corresponding to the duration of the SATPOS file (for a specific station and a specific satellite). This file can be non chronological if the equator is inside the acquisition area of the station. It can be time-sorted with the unix command sort.

The command **ephe** is activated with the name of the files satpos\_noaxx\_yyyymmdd.txt and ephe\_noaxx\_yyyymmdd.txt.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

Usage is:

**ephe** <satpos\_file> ephemeris\_file

#### 4.3.25. Description of the command TRACKING

(See also the reference manual man pages *tracking.1*)

For an antenna, the command **tracking** creates a file of angles and tracking from a SATPOS file (for all the orbits included and those which can be acquired). Ephemeris messages are directed to the standard output and possibly to an ephemeris file.

The command **tracking** is activated with the name of the files satpos\_noaxx\_yyyymmdd.txt and ephe\_noaxx\_yyyymmdd.txt.

Usage is:

**tracking** <satpos\_file> ephemeris\_file

#### 4.3.26. Description of the command ANTCNFT

(See also the reference manual man pages *antcnft.1*)

The command **antcnft** updates the ephemeris file and indicates if there are orbital tracking conflicts for a given antenna.

The command **antcnft** is activated with the name of the I/O file ephe\_yyyymmdd.txt.

Usage is:

**antcnft** < ephemeris\_file> ephemeris\_file

#### 4.3.27. Description of the command DRIFTEPHE

The command **driftephe** updates the ephemeris file.

It is activated with the name of the I/O file ephe\_yyyymmdd.txt.

Usage is:

**driftephe** < ephemeris\_file> ephemeris\_file

#### 4.3.28. Description of the script TBUSDISP

(See also the reference manual man pages *tbusdisp.1*)

The script **tbusdisp** is activated after the read of 3 arguments (interactive questions/answers):

- \* Satellite name (or end)
- \* Search method (nearest or last\_preceding, default=nearest)
- \* Date dd/mm/yy or dd/mm/yy hh:mm:ss.sss

#### 4.3.29. Description of the script TLEPRINT

(See also the reference manual man pages *tleprint.1*)

The script **tleprint** is fully interactive questions/answers:

1. enter 2-line bulletin filename
  2. enter satellite name or end
- until word "end" is entered

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.3.30. AVHRR and HIRS level 1b file verification : PRHAVH and PRHIRS

Usage is :

```
prhavh -s sss -e eee filename
prhirs -s ssss -e eee filename
  -s sss : starting avhrr/hirs scan line
  -e eee : ending avhrr/hirs scan line
  filename : file to look at
```

The script generates dynamically the user input options files *prhavh.inp* , *prhirs.inp* .

The program is then launched with the user options files as input.

```
prhavh.exe < prhavh.inp
prhirs.exe < prhirs.inp
```

Lastly, it deletes the input files *prhavh.inp*, *prhirs.inp* .

#### 4.3.31. MSU level 1b file header verification PRHMSU

Usage is :

```
prhmsu filename
  with filename : file to look at
```

The script generates dynamically the user input options file *prhmsu.inp* .

The program is then launched with the user options file as input.

```
prhmsu.exe < prhmsu.inp.
```

Lastly, it deletes the input file *prhmsu.inp* .

#### 4.3.32. DCS level 1b file verification PRHDCS

Usage is :

```
prhdcs -s sss -e eee filename
  -s sss : starting dcs line
  -e eee : ending dcs line
  filename : file to look at
```

The script generates dynamically the user input options files *prhdcs.inp*.

The program is then launched with the user options files as input.

```
prhdcs.exe < prhdcs.inp
```

Lastly, it deletes the input files *prhdcs.inp*.

#### 4.3.33. Source file identification: HRPTIDF

Usage is :

```
hrpidf [-Y yyyy] [-s] [-d] [-h] [-n] [-i] source
```

The script **hrpidf** can be activated with basic information parameters (**-s -d -h -n**).

The **-i** option provides all the basic information about the source (hrpt format) in only one call.

Examples:

```
hrpidf -i hrpt_noaa1419961121_0036_09757.hrp returns noaa14 19961121 0036 09757
```

```
hrpidf -s hrpt_noaa1419961121_0036_09757.hrp returns only noaa14
```

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.3.34. Level 1b products identification: L1BIDF

Usage is :

**l1bidf** [-s] [-d] [-h] [-n] [-t] [-i] source

The **-t** option provides data type of the source in 1b format.

The **-i** option provides all the basic information about the 1b format source in only one call.

Examples:

**l1bidf** -i dcs11b-noaa1419961121\_0036\_09757.11b

returns noaa14 19961121 0036 09757 dcs cms

**l1bidf** -t dcs11b-noaa1419961121\_0036\_09757.11b

returns only dcs

#### 4.3.35. Level 1c products identification: L1CIDF

Usage is :

**l1cidf** [-s] [-d] [-h] [-n] [-t] [-i] source

The **-t** option provides data type of the source in 1c format.

The **-i** option provides all the basic information about the 1c format source in only one call.

Examples:

**l1cidf** -i hirs11c-noaa1419961121\_0036\_09757.11c

returns noaa14 19961121 0036 09757 hirs cms cms

**l1cidf** -t hirs11c-noaa1419961121\_0036\_09757.11c

returns only hirs

#### 4.3.36. Level 1d products identification: L1DIDF

Usage is :

**l1didf** [-s] [-d] [-h] [-n] [-t] [-i] source

The **-t** option provides data type of the source in 1d format.

The **-i** option provides all the basic information about the 1d format source in only one call.

Examples:

**l1didf** -i hirs11d-noaa1419961121\_0036\_09757.11d

returns noaa14 19961121 0036 09757 hirs cms cms

**l1didf** -t hirs11d-noaa1419961121\_0036\_09757.11d

returns only hirs

#### 4.3.37. Write out a message: LIBLOG

Usage is :

**Log\_xxxx** "text of the message"

With xxxx is the type of the message :

info notice critical

warning debug emergency

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

error            text

Examples :

**log\_info** "start of processing \${FILE}"

**log\_error** "file should be given with a full path name"

#### 4.3.38. Get the orbit number: SDH2ORBNUM

**sdh2orbnum.ksh** allows to get the orbit number for a NOAA satellite and for a given instant.

Usage is :

**sdh2orbnum -s** satid **-d** yyyyymmdd **-h** hhmm

Executable called : **sdh2orbnum.exe**

#### 4.3.39. Decode 1c BUFR files: AAPP\_DECODEBUFR\_1C or ECCODES\_DECODEBUFR\_1C

Usage is:

**aapp\_decodebufr\_1c [-i files] [-v] [instruments]**

**eccodes\_decodebufr\_1c [-i files] [-v] [instruments]**

where *files* is a list of one or more input files. Defaults to "hrsn.bufr aman.bufr ambn.bufr mhsn.bufr iasi.bufr".

If the -v option is present, the first observation is printed out in full.

The optional "instruments" argument is provided in case you have a BUFR file that contains more than one instrument and you want to specify which one to extract.

This routine calls the ECMWF BUFRDC or ecCodes library. In the case of BUFRDC, it uses the BUFR tables in directory \$BUFR\_TABLES. The BUFR tables are selected automatically according to the value of the Originating Centre and Sub-centre in Section 1 of the BUFR message. Please see the script for details.

#### 4.3.40. Encode 1c BUFR files : AAPP\_ENCODEBUFR\_1C or ECCODES\_ENCODEBUFR\_1C

Usage is:

**aapp\_encodebufr\_1c [-N | -n] [-i files] [-s skip\_scans] [-v nonelyes] instruments**

**eccodes\_encodebufr\_1c [-N | -n] [-i files] [-s skip\_scans] [-v nonelyes] instruments**

where *instruments* is a list of instruments, from the list: "HIRS AMSU-A AMSU-B MHS IASI ATMS CRIS HIRS1D AMSUB1D IASI1D ATMS1D CRIS1D"; *files* contains the input file names for each instrument. Defaults to "hrsn.11c aman.11c ambn.11c mhsn.11c iasi.11c atms.11c cris.11c hirs.11d amsub.11d iasi.11d atms.11d cris.11d". The -N flag causes NEDT values to be included for ATOVS only (default is to include them), suppressed by -n. The -v flag controls whether a software version number is to be included; default is yes for ATOVS and none for other instruments.

There are several environment variables that can be used to fine-tune the BUFR encoding, e.g. to specify your Originating Centre ID. Please see the script for details.

This routine calls the ECMWF BUFRDC or ecCodes library. In the case of BUFRDC, it uses the BUFR tables in directory \$BUFR\_TABLES.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

Note that the BUFR sequences for level 1d have several Met Office local descriptors; they are primarily intended for use either within the Met Office or by NWP Centres that use the Met Office's Unified Model.

#### 4.3.41. Decode Sensor Data Record files for ATMS, CrIS, MWTS, MWHS, MWTS2, MWHS2, IRAS, HIRAS

Usage is :

```

atms_sdr [-o Outputfile] [-g Geofile] SDRfile [TDRfile]
cris_sdr [-o Outputfile] [-g Geofile] [-H] [-B] [-N] SDRfile'
mwts_sdr [-o Outputfile] SDRfile
mwhs_sdr [-o Outputfile] SDRfile
mwts2_sdr [-o Outputfile] SDRfile
mwhs2_sdr [-o Outputfile] SDRfile
iras_sdr [-o Outputfile] SDRfile
hiras_sdr [-o Outputfile] SDRfile

```

These tools convert the SDR files for ATMS, CrIS, MWTS, MWHS, IRAS and HIRAS into AAPP 1c format. They require AAPP to have been built with the HDF5 library.

For ATMS and CrIS, which have separate geolocation files, the user is able to specify the geolocation file explicitly. However, this is mainly useful for pre-launch test data and would only be necessary for operational data if the attribute "N\_GEO\_Ref" is missing or invalid.

The ATMS 1c format has space for both antenna temperatures and brightness temperatures. If required, the antenna temperatures may be read from a TDR file. However, most users will not need to do this.

The MWTS and MWHS tools (for the sounders on the Chinese FY-3 satellites) include some quality checking – including scan-to-scan consistency of the calibration slope; geolocation reasonableness test; antenna position check. The intention is that only reliable brightness temperatures will appear in the output 1c files.

#### 4.3.42. FY-3 mapping tools: mwhs\_to\_mwts, mwhs2\_to\_mwts2, mwts2\_to\_mwhs2, mwts2\_to\_iras, mwhs2\_to\_iras

Usage (example) :

```
mwhs_to_mwts mwts_file mwhs_file
```

*i.e. program target\_file source\_file*

The tool maps the MWHS brightness temperatures to the MWTS grid and stores the results in the MWTS level 1c file. It uses the latitude/longitude information from the two files, i.e. it does not use pre-defined look-up tables.

For each MWTS spot, all MWHS spots are identified that are within a specified angular tolerance from the MWTS spot (tolerance specified in the source code). Then either the *median* brightness temperature is computed and stored for each channel, or the nearest neighbour brightness temperature is used.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

In the case of mapping MWHS to MWTS, the median is always used, because this method was found to be robust when there are corrupt MWHS BTs (which were observed from time to time when the data were first distributed by EUMETSAT). Note that the MWHS beam width is much narrower than that of MWTS, so there will be many MWHS footprints within a MWTS footprint.

In the case of MWTS2 and MWHS2 (on FY-3C), the median is used only if the appropriate environment variable is set: MWHS2\_USE\_MEDIAN or MWTS2\_USE\_MEDIAN. By default, the nearest neighbour is used. MWTS2 and MWHS2 footprints are much more similar in size than is the case for MWTS and MWHS, therefore nearest neighbour mapping is usually more appropriate.

To map both MWTS2 and MWHS2 to IRAS, run the programs *mwts2\_to\_iras* and *mwhs2\_to\_iras* sequentially. The IRAS 1c format has space for 28 mapped channels: 13 MWTS2 followed by 15 MWHS2.

#### 4.3.43. **is-mmam.exe**

The command **is-mmam.exe** verifies if a PFS 10 file (HKTM) or a CCSDS file includes a MMAM message

Usage is:

**is-mmam.exe** [ -ccsds <ccsds file>| -pfs10 <pfs10\_file> ]

*example :*

```
$ is-mmam.exe -ccsds apid6.ccsds
TRUE
```

#### 4.3.44. **mmam-main.exe**

The command **mmam-main.exe** extracts a MMAM compressed bz2 file from a PFS 10 file (HKTM) or a CCSDS file.

Usage is :

**mmam-main.exe** [ -ccsds <ccsds file>| -pfs10 <pfs10\_file> ] <bz2\_file>

*example :*

```
$ mmam-main.exe -ccsds apid6.ccsds mmam.bz2
```

#### 4.3.45. **print-mmam-obt-utc.pl**

The script **print-mmamm-obt-utc.pl** extracts the OBT UTC correlation parameters (utc0 ccu-obt-0 clock-step) from a MMAM message and prints them.

Usage is :

**print-mmam-obt-utc.pl** <MMAM\_file>

*example :*

```
$ print-mmam-obt-utc.pl MMAM_GENERATED_M02_215_20120612081404.xml
2012-06-12T07:02:58.285 2677315586 3906239944
```



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.3.46. patch-level0-from-mmam.exe

The command **patch-level0-from-mmam.exe** changes the VIADR records in a PFS level0 with OBT UTC correlation parameters (utc0 ccu-obt-0 clock-step)

Usage is :

**patch-level0-from-mmam.exe** utc0 ccu-obt-0 clock-step ....\_xxx\_00\_...  
 utc0 ccu-obt-0 clock-step : parameters as they are printed by print-mmam-obt-utc-.pl

*example :*

```
$patch-level0-from-mmam.exe 2012-06-12T07:02:58.285 2677315586 3906239944 ¥
AVHR_P13_00_M02_20120612084401Z_20120612085256Z_N_0_20120612085410Z
```

#### 4.3.47. atms1c\_print\_nedt

The command **atms1c\_print\_nedt** prints a table of ATMS NEΔT values for warm and cold calibration views. The mean and standard deviation are displayed, for each channel and view.

Usage :

**\$ atms1c\_print\_nedt** <ATMS 1c file>

#### 4.3.48. aapp1c\_quicklook.py

A Python tool for quick-look display of brightness temperature (microwave instruments and HIRS/IRAS), radiance (CrIS) or scaled radiance (IASI), for l1c.h5 or l1d.h5 files. Prerequisites: numpy, matplotlib, h5py, cartopy.

Usage: **aapp1c\_quicklook.py** [-h] -i INPUTFILE [-c CHANNEL] [-r LATLONRANGE] [-s SYMSIZE] [-nd] [-ns] [-a] [-H]

optional arguments:

```
-h, --help                show this help message and exit
-i INPUTFILE, --in INPUTFILE
                          input h5 file
-c CHANNEL, --channel CHANNEL
                          Channel to display, or 'surface' or 'rain'
-r LATLONRANGE, --range LATLONRANGE
                          geographical limit: 'x0 x1 y0 y1', default
                          '-180 180 -90 90'
-s SYMSIZE, --symsize SYMSIZE
                          symbol size, default 4
-nd, --no_display         don't display image on screen
-ns, --no_save            don't save image
-a, --auto                auto range for lat/lon
-H, --high_resolution    high resolution coastline
```

### 4.4. MAIA

#### 4.4.1. Introduction

MAIA is software providing cloud detection and classification at pixel resolution for both VIIRS and AVHRR imagers. In AAPP8, version 4 of MAIA replaces all the previous versions of MAIA (MAIA2 and MAIA3) which existed in the previous versions of AAPP. Users can use high level scripts to produce the MAIA cloud products at the imager resolution (**MAIA4\_RUN** for VIIRS and **MAIA4\_RUN\_AVHRR** for AVHRR). All cloud products are gathered in one file. This file is named

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

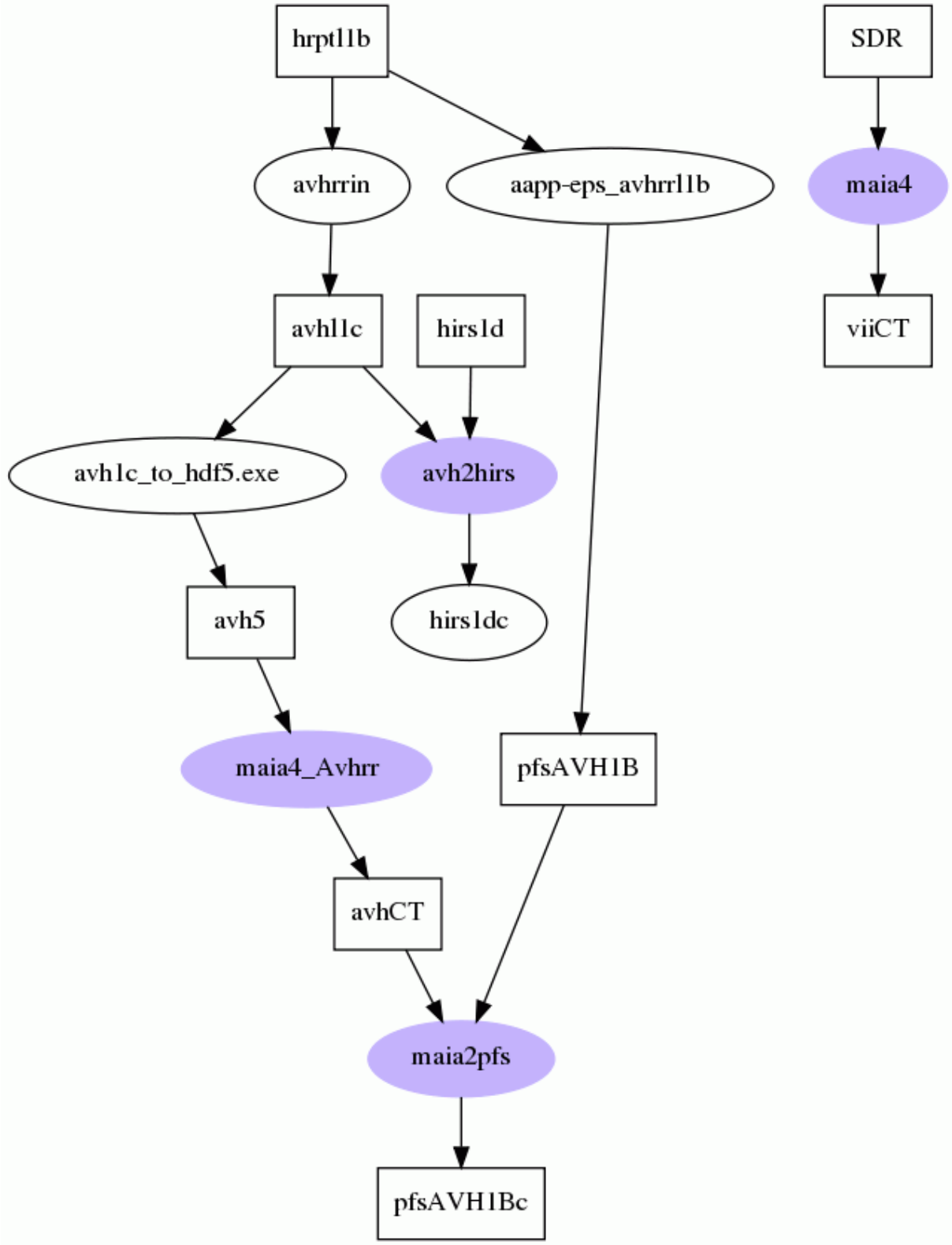
“**avhCT**” for the AVHRR cloud products and “**viiCT**” for the VIIRS cloud products. The format of these files is hdf5 (Hierarchical Data Format version 5) and the detailed documentation of their datasets can be found in the AAPP Data Formats document. These files contain (non exhaustive list):

- The Cloud Mask
- The Cloud Mask Confidence
- The Cloud Top Pressure
- The Cloud Top Temperature
- The Cloud Type
- The Sea Surface Temperature.

The algorithms used by MAIA are described in the document MAIA version 4 Scientific User Manual [39]

Furthermore, users can use tools: **avh2hirs** and **maia2pfs.exe**, to add cloud information in the field of view of ATOVS sounders (with avh2hirs) or IASI (with maia2pfs.exe and OPS-LRS). This functionality is also available for VIIRS/CrIS (see AAPP document VIIRS to CrIS Mapping [38])

**4.4.2. MAIA interfaces**



**Figure 4-29 : MAIA interfaces**

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.4.3. Usage

MAIA requires several resource files to get prior information on the state of the atmosphere and the surface. MAIA needs numerical weather prediction (NWP) model fields. The location of the NWP model fields can be defined with the DIR\_FORECAST environment variable. The format of the NWP model fields has to be GRIB. The ECMWF ecCodes or the ECMWF grib-api packages are used for reading GRIB files. It is possible to use GFS files as NWP. In this case MAIA can download directly the forecast files.

##### AVHRR :

---

MAIA4\_RUN\_AVHRR is the high level script for MAIA.

Usage is :

**MAIA4\_RUN\_AVHRR [hrpt1b|AVHRR PFS L0|AVHRR PFS L1B**

MAIA4\_RUN\_AVHRR accepts the avhrr AAPP level 1b format, as well as the AVHRR METOP PFS format (L0 from a Direct Broadcast reception, or L1B from the EUMETCast reception).

MAIA4\_RUN\_AVHRR reads ATOVS\_ENV to get the conditions of the run, converts the input file in AVHRR level 1c in hdf5 format and calls maia4\_Avhrr

maia4\_Avhrr.ksh is the low level script call by MAIA4\_RUN\_AVHRR

usage is:

**maia4\_Avhrr -s satimg -d yyymmdd -h hhmm -n nnnnn -o outputfile avh5**

avh5 : is the name of the AVHRR level 1c file converted in hdf5 by convert\_to\_hdf5

maia4\_Avhrr uses the korn shell functions of maia\_env.ksh and maia\_date.ksh and creates the execution environment for maia\_Avhrr.exe. It creates symbolic links with ancillary data files (atlas, thresholds, climatology SST files) and with the forecast files. Then, it calls maia\_Avhrr.exe.

maia\_Avhrr.exe is the main executable for AVHRR data.

The symbolic links are removed after the execution on this program and the output file is renamed.

If the user does not give an output name, a standard output name is given which follows the syntax described in the document MAIA version 4 Scientific User Manual [39]

Exemple:

avhCT\_N19\_DB\_20170515\_S130128\_E131234\_ASC\_D\_La055\_Lo0019\_4266.h5

##### VIIRS :

---

**MAIA4\_RUN is the high level script for MAIA**

Usage is:

**MAIA4\_RUN viirs\_sdr\_directory**

viirs\_sdr\_directory is the directory containing the VIIRS SDR files to be processed.

maia4.ksh is the low level script call by MAIA4\_RUN

usage is:

**maia4 <sdr\_geolocation\_file>**

sdr\_geolocation\_file is a M geolocation SDR granule (GMODO or GMTCO file). The I SDR and M SDR granules are supposed to be found in the same directory as the geolocation granule.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

maia4.ksh uses the korn shell functions of maia\_env.ksh and maia\_date.ksh and creates the execution environment for maia\_Viirs.exe. It creates symbolic links with ancillary data files (atlas, thresholds, sst file) and with the forecast files. Then, it calls maia\_Viirs.exe.

**maia\_Viirs.exe** : main executable for VIIRS data

The symbolic links are removed after the execution on this program and the output file is renamed. A standard output name is given which follows the syntax described in the document MAIA version 4 Scientific User Manual [39]

Exemple :

viiCT\_npp\_DB\_20121010\_S132824\_E132947\_ASC\_D\_La050\_Lo-012\_0001.h5

maia\_Viirs.exe requires M SDR granule files and uses, if present, VIIRS I SDR granule files.

**Maia\_Viirs.exe cannot process VIIRS SDR aggregate files.**

#### 4.4.4. USER INPUT PARAMETERS FOR MAIA4 IN ATOVS\_ENV

DIR\_FORECAST = source directory of the forecast

The forecast file default pattern is

YYYYMM/YYYYMMDDHHMN.ECH

surface constant file: YYYYMM/YYYYMMDDHHMN.CST (GRIB with parameters Geometrical height and Land-sea mask)

MAIA4\_USE\_GFS= if equal to "yes" GFS is used

MAIA4\_REMOTE\_GFS\_DIR=URL where GFS files can be downloaded default value is "http://jpsdb.ssec.wisc.edu/cspp\_v\_2\_0/ancillary"

If MAIA4\_USE\_GFS="yes" the forecast file pattern is :

YYYY\_MM\_DD\_CCC/gfs.press\_gr.0p5deg\_pt.YYYYMMDD\_HH\_ECH.npoess.grib2

Those files are downloaded from MAIA4\_REMOTE\_GFS\_DIR if not present in the

DIR\_FORECAST directory.

DIR\_FORECAST can be common with the CSPP EDR ancillary data directory

(\${CSPP\_EDR\_HOME}/anc/cache).

NFORPERDAY= number of forecasts per day (2 or 4, 4 by default)

DIR\_MAIA4\_THRESHOLDS= directory of the MAIA4 thresholds by default :

\${AAPP\_PREFIX}/AAPP/data\_maia4/thresholds

DIR\_MAIA4\_ATLAS=directory of the MAIA4 atlas + topography files by default :

\${AAPP\_PREFIX}/AAPP/data\_maia4/atlas

PAR\_MAIA4\_COMPRESS compression of viiCT or avhCT files (0 : no compression, 1 compression)

*MAIA4 box sizes :*

MAP\_BOX\_PSIZE= box size for environment in pixel (default value : 16)

MAP\_BOX\_LSIZE= box size for environment in line (default value : 16)

LOCAL\_BOX\_NPB= local box size for variance in pixel (default value : 3)

LOCAL\_BOX\_NLB= local box size for variance in line (default value : 3)

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**4.4.5. MAIA Software components**

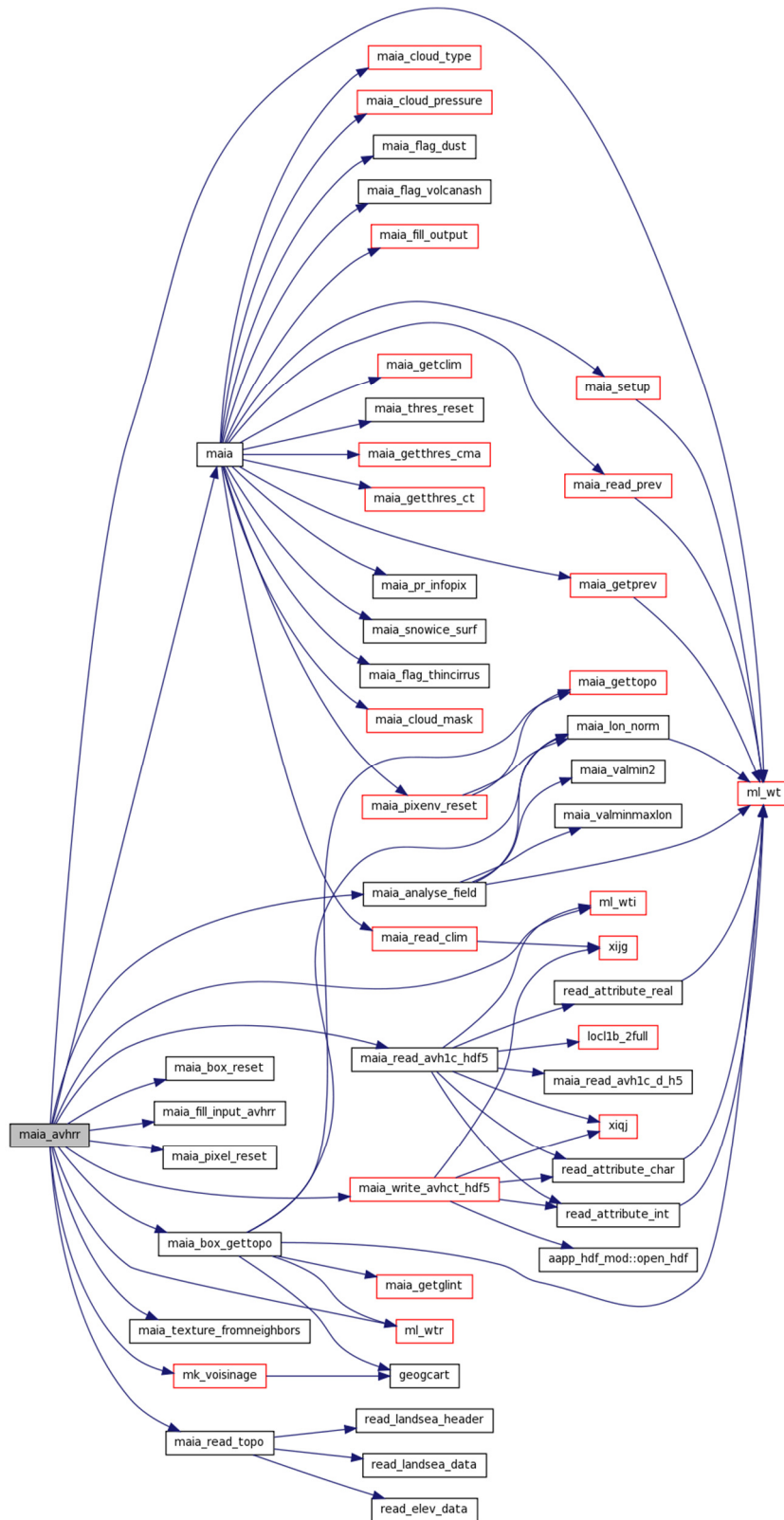


Figure 4-30 : MAIA components for AVHRR

(MAIA components for VIIRS are identical except for the read and write components)

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.4.6. MAIA file source codes description

**AAPP/src/maia4/bin directory :**

**maia\_Viirs.F90:** main program for VIIRS

**maia\_Avrr.F90:** main program for AVHRR

**MAIA4\_RUN.ksh:** main script for running MAIA on all VIIRS SDR granule in a directory, this script call the maia4.ksh script for each M geolocation SDR granule present in the input directory.

**maia4.ksh:** low level script for running MAIA (described in section 4.4.3)

**MAIA4\_RUN\_AVHRR.ksh:** main script for running MAIA on AVHRR data.

**maia4\_avhrr.ksh:** low level script to be run on a avh level 1c file hdf5 format.

**maia\_env.ksh :** script for initialising maia4 environment. Contains the following functions :

```
function get_forecast_file
function get_climatology_file
function maia4_env
function remove_maia4_env
```

**maia\_date.ksh :** provides functions for date computations

**read\_maiaCT.F90 :** this program is an example of how to read with the aapp\_viirs API the maia cloud mask.

Usage is :

```
read_maiaCT.exe viiCT_file
```

This program creates 3 files :

```
fort.20 : with longitude latitude cloud_mask
fort.21 : with longitude latitude mask_confidence
fort.23 : with longitude, latitude, cloud_type, l,p
```

**naggmaia4.py :** a tool to merge together the MAIA4 hdf5 files for different granules

**Libraries :**

**AAPP/src/maia4/libmaia4**  
Contains the core of MAIA

AAPP/src/maia4/libmaia4IO



<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

Contains input and output subroutines for MAIA, VIIRS and AVHRR.

**Ancillary files :**

The ancillary files are in the AAPP/data\_maia4 directory  
 The thresholds directory contains the different threshold files and the sst file.  
 The atlas directory contains the atlas files.

**4.5. MAIA2PFS.EXE**

maia2pfs.exe: is a program to update the CLOUD\_INFORMATION record in AVHRR PFS L1B FILES

It takes three arguments:

- Level 1b PFS file, converted to PFS format by aapp-eps\_avhrr11b
- AAPP maia4 hdf5 file format
- output level 1b PFS file with cloud mask.

Input files (maia and 1b PFS) should be the same for start and end times, a succinct control is done (line numbers and start date/time)

AVHRR PFS PFS L1B files with updated CLOUD\_INFORMATION can be then processed by OPS-LRS with IASI PFS L0 files. The resulting IASI L1C file will be filled with the cloud fraction in IASI FOV from AVHRR 1B CLOUD\_INFORMATION.

**4.6. VIIRS TOOLS**

This section describes the tools for handling VIIRS data, that were introduced with AAPP v7.5.

**4.6.1. Decode and concatenate Sensor Data Record granule files for VIIRS**

*Note: The tool “viirs\_paste\_sdr” was written before the release of hdf5 tool “nagg”. It is strongly recommended to use nagg, as it is appreciably faster, especially on machines with limited memory .*

Usage is :

**viirs\_paste\_sdr.exe < viirs\_paste\_sdr.in**

where viirs\_paste\_sdr.in contains the name of the output hdf5 file, followed by the names of all SDR VIIRS granule (M or I or DNB) to be read and concatenated and followed by the word “compress” or “uncompress”.

**viirs\_paste\_sdr.exe** creates an hdf5 file with the same structure of the original VIIRS SDR files where all channels are present and all granules are concatenated.

If present, the scale factors for brightness temperature or reflectance are applied.

viirs\_paste\_sdr.exe must be called separately for either I, M or DNB channels.

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

This tool require AAPP to have been built with the HDF5 Fortran library.

Example :

```
list=`ls ${input_dir}/SVI*.h5 ${input_dir}/GITCO*.h5`
outfile=viirs_i_${YYYYMMDD}_${HHMNSS}_${NNNNN}.h5

echo $outfile > viirs_paste_sdr.in
for i in $list
do
echo $i >> viirs_paste_sdr.in
done
echo compress >> viirs_paste_sdr.in
viirs_paste_sdr.exe <viirs_paste_sdr.in
```

This tool is based upon the libaapp\_viirs library.

#### 4.6.2. Decode EDR IMG granule files for VIIRS

Usage is :

**viirs\_edr\_img.exe < viirs\_edr\_img.in**

where viirs\_edr\_img.in contains :

- iopt (1,2 or 3)
- the band name (I, M, or NCC)
- input file name
- channel (optional)

The last 2 items may be repeated several times if iopt=2.

if iopt=1 : One Band, One file, List of channels

if iopt=2 : One band, One file, All channels

if iopt=3 : One band, list of [file, channel]

The program provide an ascii file named "fort.20".

viirs\_img\_edr.F90 is provided as an example of program which reads EDR IMG granules files.

Please note that this program has been tested only with the NPP pre-launch data tests with VIIRS I EDR IMG files from CLASS.

#### 4.6.3. The Fortran90 aapp\_viirs API

This library contains functions that enable you to read and write VIIRS SDR and IMG EDR files in a more user friendly way than using the hdf5 fortran90 API.

**User level subroutine:**

**subroutine viirs\_sdr\_load( bandname, x, filenames, err, channels , geolocfile, no\_geo)**

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

*Loads and pre-process all VIIRS data for a given Band according to options, returns x:  
If only one filename, it will be used for all channels  
If "channels" is present then corresponding channels will be read  
if not, all channels of given Band are read from one file or from the list of files  
If no\_geo is present and false, or not present  
geolocation file is loaded from geolocfile (if present) or from  
the root attribute N\_GEO\_Ref, but same directory as filenames(1). Then geolocation is processed.*

*Input/output :*

character(len=\*), intent(in) :: bandname ! VIIRS Band Name: I M or DNB  
type(viirs\_sdr), intent(inout) :: x  
character(len=\*), intent(in) :: filenames(:) ! Name of file (one for all or one per channel)  
integer, intent(out) :: err  
integer, optional, intent(in) :: channels(:) ! if present, the list of channels  
character(len=\*), optional, intent(in) :: geolocfile ! file name for geolocation  
logical, optional, intent(in) :: no\_geo ! if TRUE geolocation is not loaded  
logical, optional, intent(in) :: clean ! if TRUE remove unused arrays

**subroutine viirs\_sdr\_save( x, filename, err,compress)**

*Saves structure x to and HDF5 file  
meta data per granule are not written  
meta data for aggregate granule are written*

*Input/output :*

type(viirs\_sdr), intent(in) :: x  
character(len=\*), intent(in) :: filename ! Name of file (one for all or one per channel)  
integer, intent(out) :: err  
logical, optional, intent(in) :: compress

**subroutine viirs\_sdr\_info( x, nchannels, npixels, nlines, nscans, ngranules, err )**

*returns number of pixels/lines/scans/granules  
returns the real number of channels loaded*

*Input/output :*

type(viirs\_sdr), intent(in) :: x  
integer,intent(out)::err  
integer, intent(out) :: nchannels, npixels, nlines, nscans, ngranules

**subroutine viirs\_edr\_img\_load( bandname, x, filenames, err, channels , geolocfile, no\_geo, clean)**

*Loads and pre-process all VIIRS data for a given Band according to options, returns x:  
If only one filename, it will be used for all channels  
If channels is present then corresponding channels will be read  
if not, all channels of given Band are read from one file or from the list of files  
If no\_geo is present and false, or not present  
geolocation file is loaded from geolocfile (if present) or from  
the root attribute N\_GEO\_Ref, but same directory as filenames(1). Then geolocation is processed.*

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

*Input/output :*

character(len=\*), intent(in) :: bandname ! VIIRS Band Name: I M or DNB  
type(viirs\_edr\_img), intent(inout) :: x  
character(len=\*), intent(in) :: filenames(:) ! Name of file (one for all or one per channel)

integer, intent(out) :: err  
integer, optional, intent(in) :: channels(:) ! if present, the list of channels  
character(len=\*), optional, intent(in) :: geolocfile ! file name for geolocation  
logical, optional, intent(in) :: no\_geo ! if TRUE geolocation is not loaded  
logical, optional, intent(in) :: clean ! if TRUE remove unused arrays

**subroutine viirs\_edr\_img\_info( x, nchannels, npixels, nlines, nscans, ngranules, err )**

*returns number of pixels/lines/scans/granules  
returns the real number of channels loaded*

*Input/output :*

type(viirs\_edr\_img), intent(in) :: x  
integer, intent(out) :: err  
integer, intent(out) :: nchannels, npixels, nlines, nscans, ngranules  
integer :: channel, mchannels

**Other subroutines:**

**subroutine viirs\_sdr\_load\_channel( filename, bandname, channel, x, err )**

*loads "All\_Data" for a given channel/band from file HDF5  
loads root attributes  
loads aggregate attributes*

*Input/output :*

character(len=\*), intent(in) :: filename ! Name of file  
character(len=\*), intent(in) :: bandname ! VIIRS Band Name: I M or DNB  
integer, intent(in) :: channel  
type(viirs\_sdr), intent(inout) :: x  
integer, intent(out) :: err

**subroutine viirs\_sdr\_save\_channel( filename, bandname, channel, x, err )**

*Saves "All\_Data" for a given channel/band to an HDF5 file filename  
saves root attributes  
saves aggregate attributes  
saves dataproduct attributes*

*Input/output :*

type(viirs\_sdr), intent(in) :: x  
character(len=\*), intent(in) :: filename ! Name of file (one for all or one per channel)  
integer, intent(out) :: err  
logical, optional, intent(in) :: compress

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

**subroutine viirs\_sdr\_geo\_load( filename, bandname, x, err )**

*loads Geolocation for a given band from HDF5 file filename*

*Input/output :*

character(len=\*), intent(in) :: filename  
character(len=\*), intent(in) :: bandname ! VIIRS Band Name: I M or DNB  
type(viirs\_sdr\_geo), intent(inout) :: x  
integer,intent(out)::err

**subroutine viirs\_sdr\_geo\_save( filename, bandname, x, err )**

*saves Geolocation for a given band to an HDF5 file filename*

*Input/output :*

character(len=\*), intent(in) :: filename  
character(len=\*), intent(in) :: bandname ! VIIRS Band Name: I M or DNB  
type(viirs\_sdr\_geo), intent(in) :: x  
integer,intent(out) :: err  
logical, optional, intent(in) :: compress

**subroutine viirs\_sdr\_data\_proc( x, err )**

*processing of the data part of the structure viirs\_sdr\_data  
applies scaling factors according to channels*

*Input/output :*

type(viirs\_sdr\_data), intent(inout) :: x  
integer,intent(out)::err  
logical, optional, intent(in) :: clean ! if TRUE remove unused arrays

**subroutine viirs\_sdr\_geo\_proc( x, att, err )**

*processing of the geolocation structure viirs\_sdr\_geo  
calculates the TAI offset and calculates the UTC time for each scan.*

*Input/output :*

type(viirs\_sdr\_geo), intent(inout) :: x  
type(jpss\_meta\_aggregate), intent(in) :: att  
integer,intent(out)::err

**subroutine viirs\_sdr\_checkaggregate( x, y, err )**

*verifies agg\_att Y (viirs\_sdr\_agg\_att) is the same as the one contained in X (viirs\_sdr).*

*Input/output :*

type(viirs\_sdr), intent(inout) :: x  
type(jpss\_meta\_aggregate), intent(inout) :: y  
INTEGER, intent(out) :: ERR ! Error code

**subroutine viirs\_edr\_img\_load\_channel( filename, bandname, channel, x, err )**

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

*loads "All\_Data" for a given channel/band from HDF5 file filename  
loads root attributes  
loads aggregate attributes*

*Input/output :*

character(len=\*), intent(in) :: filename ! Name of file  
character(len=\*), intent(in) :: bandname ! VIIRS Band Name: I M or DNB  
integer, intent(in) :: channel  
type(viirs\_edr\_img), intent(inout) :: x  
integer, intent(out)::err

**subroutine viirs\_edr\_img\_geo\_load( filename, bandname, x, err )**

*loads Geolocation for a given band from HDF5 file filename*

*Input/output :*

character(len=\*), intent(in) :: filename  
character(len=\*), intent(in) :: bandname ! VIIRS Band Name: I M or DNB  
type(viirs\_edr\_img\_geo), intent(inout) :: x  
integer, intent(out)::err

**subroutine viirs\_edr\_img\_data\_proc( x, err, clean)**

*processing of the data part of the structure viirs\_edr\_img\_data  
applies scaling factors according to channels*

*Input/output :*

type(viirs\_edr\_img\_data), intent(inout) :: x  
integer, intent(out)::err  
logical, optional, intent(in) :: clean ! if TRUE remove unused arrays

**subroutine viirs\_edr\_img\_geo\_proc( x, att, err )**

*processing of the geolocation structure viirs\_edr\_img\_geo  
calculates the TAI offset and calculates the UTC time for each scan.*

*Input/output :*

type(viirs\_edr\_img\_geo), intent(inout) :: x  
type(jpss\_meta\_aggregate), intent(in) :: att  
integer, intent(out)::err

**subroutine viirs\_edr\_img\_checkaggregate( x, y, err )**

*verifies agg\_att Y is the same as the one contained in X, if any*

*Input/output :*

type(viirs\_edr\_img), intent(inout) :: x  
type(jpss\_meta\_aggregate), intent(inout) :: y  
INTEGER, intent(out) :: ERR ! Error code

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

### Low level subroutines :

For each fortran structure, 3 modules are automatically generated :

- definition modules ( `_def.F90` ) :  
contain the fortran structure definition.
- fortran I/O modules ( `_io.F90` ) :  
`_rh` subroutines : read HDF5  
`_writea` subroutines : write ASCII  
`_wh` subroutines : write HDF5
- memory modules ( `_mem.F90` )  
`free`: free pointers structure  
`init`: initialise structure  
`copy(x,y,...)`: copy structure y to x

Most structures contains arrays of pointers. The dimensions could be pixels/lines/scans/granules  
 The size of the dimensions are not part of the structure itself but could be easily given by the *size*  
 fortran intrinsic routine, e.g.:

```
nlines = size(x%BrightnessTemperature,2)
```

#### 4.6.4. VIIRS to CrIS mapping

Usage is :

```
viirs_to_cris [-dl-D] [-t threshold] [-b band] [-m Maia4file] [-C cluster_output_file] [-g Geofile] [-s  

  sorting_channel] CrISfile VIIRSfile
```

where

band is a VIIRS band name I or M  
 Maia4file is a VIIRS MAIA 4 HDF5 file  
 Geofile is a VIIRS geolocation HDF5 file  
 Crisfile is a CrIS AAPP level 1c/1d file  
 VIIRSfile is a VIIRS SDR HDF5 file  
 cluster\_output\_file is an output file name for the hdf5 file which contains the results of the  
 clusters analysis  
 sorting\_channel : (1 to 16) VIIRS channel number for clusters sorting (by default the channel  
 with the largest wavelength)  
 -d debug level 1  
 -D debug level 2  
 threshold is the minimum percentage of valid VIIRS pixels for mapping

For further information please refer to the document “VIIRS-CrIS mapping” [38].

The cluster output file can be merged with a corresponding CRISFSR bufr file using the tool  
`cris_bufr_add_clusters` (see 4.6.5).

<b>NWP SAF</b>	<b>AAPP DOCUMENTATION SOFTWARE DESCRIPTION</b>	Doc ID : NWPSAF-MF-UD-002 Version : 8.1 Date : March 2019
----------------	--	---

#### 4.6.5. VIIRS\_CRIS\_MAIA4\_RUN

The VIIRS\_CRIS\_MAIA4\_RUN script (introduced in AAPP v8.4) is an end-to-end script that ingests VIIRS and CrIS SDR files and generates CrIS BUFR files that include cloud parameters (from MAIA) and the VIIRS cluster analysis. The user may wish to customize this script.

The script assumes that input VIIRS files are individual granules, i.e. not aggregated by CSPP. The VIIRS granule size is ~85s so expect ~9 granules in a typical 12-minute pass.

The script runs multiple sub-processes (one instance of MAIA for each VIIRS granule), so a multi-core machine is recommended. AAPP must have been built with the hdf5 and ecCodes libraries linked in.

The following tools, external to AAPP, are required:

nagg v1.6.2 or later<sup>1</sup>  
 python (for AAPP tool *naggmaia4.py*)

Input is via environment variables:

CRIS\_SDR\_DIR - directory containing CrIS SDR files. Default: current  
 VIIRS\_SDR\_DIR – directory containing VIIRS SDR files. Default: current  
 VIIRS\_BANDS – bands to include in the cluster analysis. Default: “M5 M7 M12 M15 M16” (i.e. the AVHRR-like bands)  
 DIR\_FORECAST – directory containing GFS forecast files. You could download these in advance if you want to save time.

Products (in particular the CrIS BUFR file) are generated in the current directory.

The script carries out the following steps:

1. Creates working directories – one for each VIIRS granule
2. Downloads the forecast files appropriate to the start/end times of the data
3. Runs MAIA4\_RUN for each granule (in parallel) to generate the cloud mask
4. Merges the VIIRS SDR granules for the required bands, and the geolocation files, using *nagg*
5. Merges the MAIA4 granules using *naggmaia4.py*
6. Merges the CrIS SDR granules using *nagg*
7. Converts CrIS SDR to AAPP 11c format using *cris\_sdr*
8. Performs a channel selection using *cris\_channels*
9. Runs *viirs\_to\_cris* to transfer the cloud parameters to the CrIS 11c file and to generate a cluster file in hdf5 format.
10. Converts the CrIS 11c file to BUFR (CRISFSR), using *eccodes\_encodebufr\_1c*
11. Runs *cris\_bufr\_add\_clusters* to merge the cluster file from step 8 with the BUFR file from step 9, creating the final BUFR file

<sup>1</sup> [https://support.hdfgroup.org/projects/jpss/jpss\\_downloads.html](https://support.hdfgroup.org/projects/jpss/jpss_downloads.html)