NWP SAF

SSMIS UPP Averaging Module

Technical Description

Version 1.0

19th November 2010

SSMIS UPP Averaging Module

Technical Description

This documentation was developed within the context of the EUMETSAT Satellite Application Facility on Numerical Weather Prediction (NWP SAF), under the Cooperation Agreement dated 1st 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.

Change record

Version	Date	Author	Approved	Remarks
0.1	20/11/2009	Anna Booton	-	Initial
0.2	10/08/2010	Nigel Atkinson	-	Version for NWP SAF
0.3	16/11/2010	Anna Booton	-	Add description of UPP_ToBeAveraged namelist
1.0	19/11/2010	Anna Booton	Dick Francis	

Contents

1 Overview

1.1 Introduction

The first SSMIS instrument, on DMSP F16, was launched in 2003. The original function of the SSMIS Pre-processor (referred to in this document as *PP*) was to ingest Temperature Data Record (TDR) files in BUFR format, apply antenna temperature to brightness temperature corrections, map the different channels to a common grid and perform spatial averaging. However, after the launch of F16 it soon became apparent that significant corrections to the brightness temperatures were required, on account of solar intrusions on the calibration target and an imperfect main reflector. The *PP* software was therefore developed by the Met Office to apply these corrections and to flag bad data – see *Scientific Description* (document NWPSAF-MO-UD-014). It is important to note that the *PP* software only works with F16 data. Following the launch of DMSP F17, it became clear that a more sophisticated processing method was required, and this led to the development of the Unified Pre-processor (UPP), which combines the functionality of the Met Office *PP* software with the correction algorithms developed at the Naval Research Laboratory (NRL).

SSMIS data from DMSP F16, F17 and F18 are now being processed routinely at source with the UPP (which is maintained by NRL) and distributed to meteorological centres with the corrections already applied. However, there is still a need for a stand-alone averaging module, since different centres will have different requirements for spatial averaging.

Thus the main function of the UPP averaging module (or pre-proce0ssor), described in this document, is to:

- Average the brightness temperature data over selectable spatial scales

as described in the *Scientific Description*.

The pre-processor takes as input a UPP BUFR file, which contains one orbit's data corresponding to the 24 SSMIS channels (from the LAS, UAS, IMA and ENV instruments as described in the *Scientific Description*). Unlike that of the SSMIS pre-processor, this input data has already been remapped to the LAS grid, corrected, and observations affected by solar intrusions have been flagged accordingly. Thus, these functions are not carried out by the UPP averaging module.

The pre-processor generates as output the averaged data as a BUFR file.

A flowchart for the UPP averaging module is given in Appendix 1 and a calling tree is included in Appendix 2.

As previously described in the *PP Technical description*, an important part of the code involves setting up working arrays containing the SSMIS radiance data and associated auxiliary data. These data are stored in *f90-derived* structures. The brightness temperature data are stored in three dimensional (3D) arrays within these structures. These 3D arrays are indexed by scan line (*i.e. along track* scan line index), scan position (*i.e. across swath* field of view number) and channel number. As described in Sections 3 and 4 of the *Scientific Description*, this indexing facilitates the use of simple averaging and remapping algorithms.

Extensive use is made of UK Met Office BUFR encode and decode routines which are made available as an object library at compilation. Details on compiling this object library, the pre-processor and running the pre-processor are given in section 3.

The UPP data are initially read in for the whole orbit, and the resulting data are stored in the UPP data structure (`UPP_DATA`) for use whilst averaging the data. The averaging is then carried out by reading in the UPP data again from the original BUFR file, scan line-by-scan-line, but the averaged brightness temperature values are substituted into the BUFR *values* array before the data are re-encoded as a BUFR message and output to a new BUFR file. The averaging is carried out on brightness temperature data in the ORBIT_DATA structure using pre-computed averaging weights read in from an external ASCII file. The averaging algorithm is described in Section 4 of the *Scientific Description*. This method of averaging is identical to that carried out by the SSMIS pre-processor, described in Section 4 of the *Scientific Description*, and in the *PP Technical description*.

1.2 Input Data/Working Array

There are two groups of input data used by the pre-processors:

1. **Instrument** data comprising the brightness temperature data for all of the SSMIS channels together with related data (scan line, scan position, latitudes, longitudes, time of the observation, surface and rain flags, onboard calibration data etc). All of this data is present in the SSMIS UPP BUFR data file.
2. **Coefficient data** required by the averaging routines. This is read in from ASCII files.

The instrument data are available initially as an external binary BUFR file. There is normally one BUFR file per orbit. The BUFR file's name usually takes the form:

```
NPR_TDUP.SB_D09265_S0338_E0524_B1486869_NS
```

The "UP" component denotes that the file has been processed by the UPP. The sixth field (*e.g.* here `B1486869`) encodes the orbit number, in this case the orbit commencing revolution 14868, and ending with the beginning of revolution 14869. For example, the next orbit will have `B1486970` in the sixth field.

The filename is made available to the pre-processor via a text file (`UPP_WORKLIST`). The file `UPP_WORKLIST` can be prepared by a Unix shell script which scans an incoming directory for new BUFR files, as described in 3.5. The text file containing the name is read in by the routine `SSMIS_UPP_GetBUFRFileName`.

The first stage in processing involves reading-in the brightness temperature data, rain and surface consistency flags, into the `UPP_DATA/ORBIT_DATA` array of derived data structure (`ORBIT_DATA_TYPE`). This input is carried out in the routine `SSMIS_UPP_GetUPPdata`. The `ORBIT_DATA` array is then passed to the averaging routine.

The only coefficient data utilized by the UPP pre-processor is that of the superobbing coefficients. This data is read in from an external ASCII file (`superob_coefficients.dat)` during the averaging routine `SSMIS_UPP_Average`, and is stored in the `coefficients` data structure (of derived data structure (`SUPEROB_COEFFS_TYPE)`).

An additional namelist file, `UPP_AUXILIARY.nl` is also required by the UPP pre-processor. This file contains additional namelists, providing the user with extra control over the data processing.

1.3 Averaging data

The brightness temperature data for the entire orbit are averaged using the algorithm described in Section 4 of the *Scientific Description*, and in an identical manner to that of the SSMIS pre-processor, described in the *PP Technical Description*. The implementation involves reading in the UPP data from the BUFR file, again scan line-by-scan line, and copying the averaged brightness temperatures (from the `ORBIT_DATA` structure) into the relevant part of the decoded BUFR message (*i.e.* to the appropriate slot in the *values* array). The averaged data are output to a BUFR file in exactly the same format as the original input BUFR file, except that the brightness temperatures, rain and surface flags have been modified as a result of the averaging. If required, the user can retain the original brightness temperature values for selected channels, rain or surface flags by specifying the appropriate switches and parameters in the `UPP_AUXILIARY.nl file.`

The Message Subtype (in Section 1 of the output BUFR files) is set to 7, to allow the output files to be distinguished from the input files (subtype 3) and from the output files of the *PP* package (subtype 6 for averaged data and subtype 5 for unaveraged data).

1.4 Output diagnostic files

Currently, no diagnostic files are output.

1.5 Intrusion mapping, Remapping of data, and Correcting for Reflector Emission

The intrusion mapping, data remapping, and the reflector emissions correction functions utilized by the SSMIS pre-processor (as described in the *PP Technical Description*) are no longer required for processing of the (already remapped and corrected) UPP data. Thus these routines are not included in the UPP averaging module.

2 Description of main subroutines

2.1 Main Program and top level subroutine

The main program `SSMIS_UPP_PREPROCESSOR` is comprised of four main components (numbered in accordance with the numbering scheme used throughout the code). These are:

1. The name of the UPP BUFR file to be processed is obtained.
2. The UPP data is read in for the entire orbit and is saved in the UPP_DATA structure.
3. The system date and time information is read in.
4. The brightness temperature data are averaged and saved as a BUFR file.

A flowchart summarizing these processes is included in Figure 1, (and Appendix 1) whilst a more detailed description of each process in included below.

It should be noted that the main program has no input or output arguments, but that the required constants are declared in the companion `SSMISMod_UPP` module and `UPP_AUXILIARY.nl` namelist file.

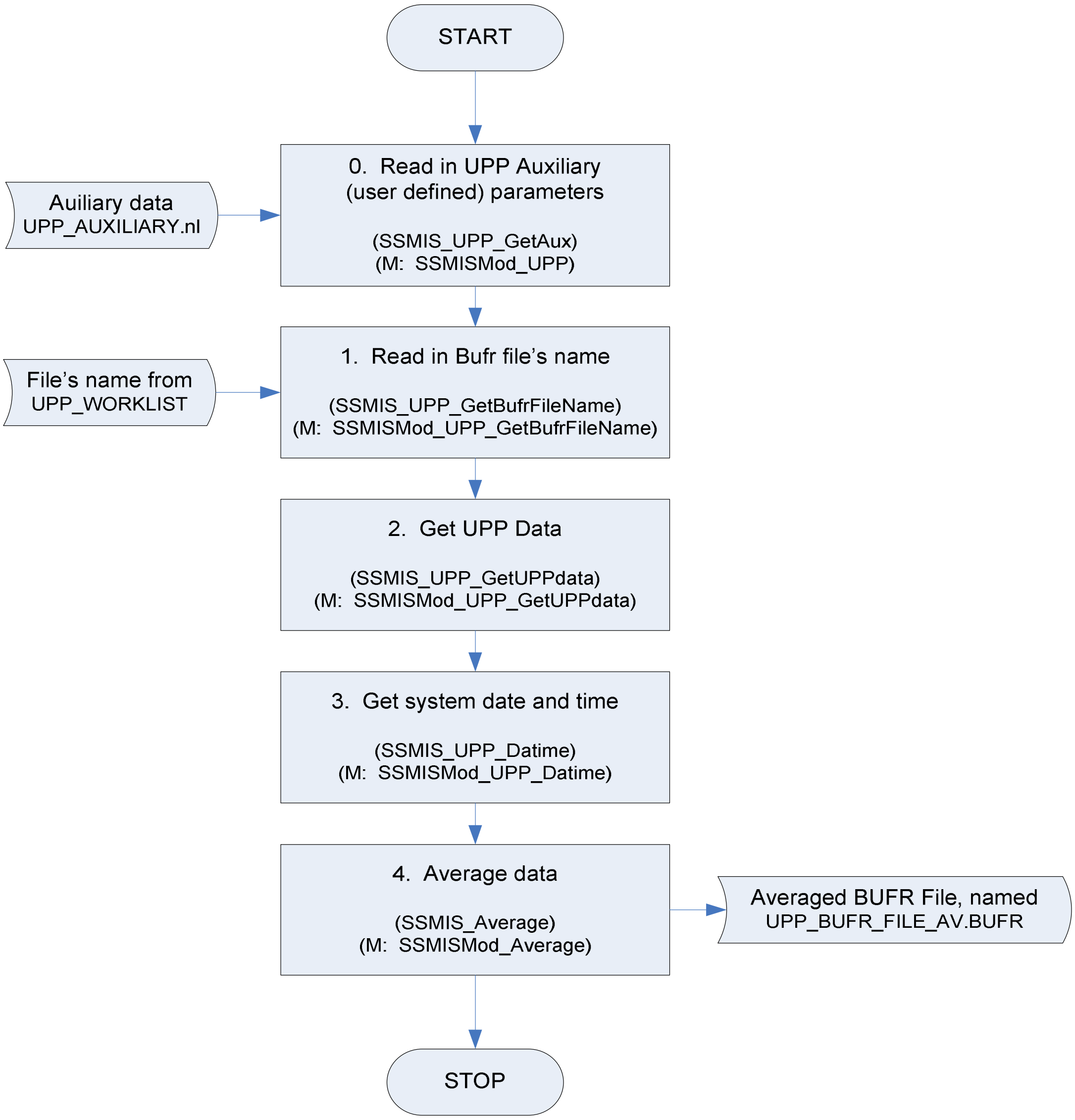


Figure 1: Flowchart showing the main program, SSMIS_UPP_PREPROCESSOR, components. The subroutines called and the containing modules (denoted by M) are included in brackets for clarity. The numbering scheme directly refers to that within the code.

2.1.1 The program structure

The main components of the UPP pre-processor, outlined below, are contained within a series of seven modules and a namelist. Therefore, the following are required by the pre-processor:

The main program:	SSMIS_UPP_PREPROCESSOR.f90
UPP pre-processor modules:	SSMISMod_UPP.f90
	SSMISMod_UPP_GetBUFRFileName.f90
	SSMISMod_UPP_GetUPPdata.f90
	SSMISMod_UPP_Datime.f90
	SSMISMod_UPP_Average.f90
	SSMISMod_UPP_DoAveraging.f90
	SSMISMod_UPP_Encode.f90
UPP namelist file:	UPP_AUXLIARY.nl

A top-level flow chart is shown in Figure 1.

2.1.2 Reading in the auxiliary data: SSMIS_UPP_GetAux

The purpose of the UPP_AUXILIARY.nl namelist file is to allow the user to easily run the pre-processor with user-defined set-up, through simply editing the required namelist parameters (rather than the code itself). The namelists within the UPP_AUXILIARY.nl file are read in by calling the subroutine SSMIS_UPP_GetAux (component 0), which is contained within the SSMISMod_UPP module. It is not compulsory to set the parameters within the namelists in this file because all switches and parameters have appropriate default values assigned in the code. However, it is necessary to include the namelists themselves (see Appendix 3). Currently the only namelist contained within this file is the 'UPP_ToBeAveraged' namelist.

2.1.2.1 The namelist: UPP_ToBeAveraged

The 'UPP_ToBeAveraged' namelist specifies the control settings for the data that are to be encoded in the UPP Averaged BUFR file. There are three control settings available to the user.

The first, 'ChannelsToBeAveraged', enables the user to choose which channels should be averaged, with those channels *not* selected retaining the original brightness temperatures.

The others, 'SetSurfaceDiscrepancy' and 'SetAveragedRainFlags' enable the user to retain the original observations' surface and rain flags respectively. When running the averaging routine, the rain flags are modified if more than 10% of the observations within the averaging domain are flagged as rain. Similarly, if there are surface discrepancies within the averaging domain the surface type is set to 'coast'. If the user does not wish to use these modified flags, then they can set them to

```
      SetSurfaceDiscrepancy = .FALSE.
```

and

```
      SetAveragedRainFlags = .FALSE.
```

accordingly.

If the user does not specify values for the above parameters by including the following code in the `UPP_AUXILIARY.nl`,

```
&UPP_ToBeAveraged
/
```

then by default, *all* channels will be averaged and the modified rainflags and surface discrepancies will be switched *on* (i.e. `SetAveragedRainFlags = .TRUE.` and `SetSurfaceDiscrepancy = .TRUE.`). For further details regarding the namelist code, see Appendix 3.

2.1.3 Reading in the BUFR file's name: SSMIS_UPP_GetBUFRFileName

The name of the UPP BUFR file, containing the data that is to be processed, is obtained. This is achieved by calling the subroutine, `SSMIS_UPP_GetBUFRFileName` (component 1), which is contained within the module, `SSMISMod_UPP_GetBUFRFileName`. This subroutine reads in the string (BUFR file's name) contained in the `UPP_WORKLIST` file. This parameter is stored by the variable `Bufr_filename`, within the module. If the filename is not of the correct UPP format (i.e. does not contain "TDUP" within the name), outlined above, execution of the pre-processor is halted.

2.1.4 Reading in the UPP data from the BUFR file: SSMIS_UPP_GetUPPdata

The UPP data are read in for the entire orbit in preparation for later use. This is accomplished by calling the `SSMIS_UPP_GetUPPdata` subroutine (component 2), contained within the module, `SSMISMod_UPP_GetUPPdata`. The flowchart of the functions carried out by this subroutine is outlined in Figure 2.

Firstly a BUFR message is read in turn from the BUFR file by a call to the subroutine `SSMIS_UPP_ReadUPPScLn` (which reads the BUFR file named into the `Bufr_filename` variable`)`, contained within the `SSMISMod_UPP` module. The message is then decoded by calling the `DEBUFR` routine. The resulting data are returned in *descriptor* (containing code numbers associated with descriptors of the data) and *values* array form. For each observation, the following data are then stored in the `UPP_DATA` array of *f90* derived data structure (`ORBIT_DATA_TYPE`), for subsequent processing:

- Scan line,
- Surface code
- Rain flags associated with the data
- Brightness temperatures

In the case of the UPP data, there are 60 fields of view/observations per scan line.

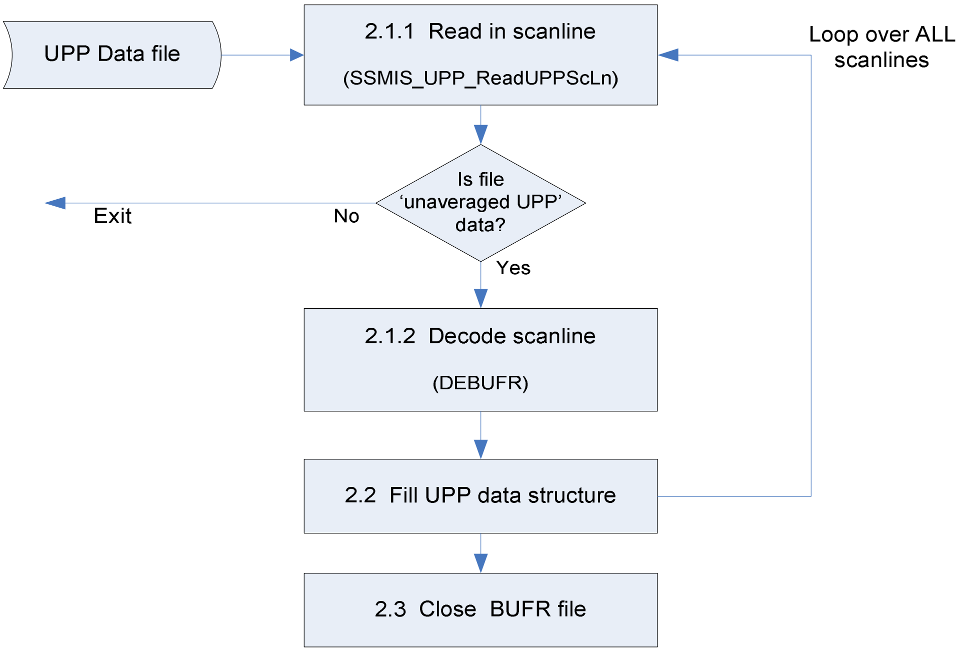


Figure 2: Flowchart showing the stages involved when calling the SSMIS_UPP_GetUPPdata, subroutine. The subroutines called are included in brackets for clarity. The numbering scheme directly refers to that within the code.

Since the BUFR message is decoded scan line-by-scan line, each line is read in turn, as described above, until all the scan lines within the BUFR file have been extracted. Following this, the number of scan lines is calculated and returned, along with the now 'filled' `UPP_DATA` structure. (When returned, it is passed to the main program as `ORBIT_DATA`).

It should be noted that the BUFR message will only be decoded if it is 'un-averaged UPP' data, as indicated by the BUFR message subtype (i.e. is subtype 3) and contains less than 5000 scan lines.

2.1.5 Reading in the date and time: SSMIS_UPP_Datime

The system date and time is retrieved by calling the `SSMIS_UPP_Datime` subroutine (component 3), contained within the `SSMISMod_UPP_Datime` module. Based upon the SSMIS PP subroutine, datim.f90, this routine returns the system date and time, and stores them in the `DATIME` array declared in the module.

2.1.6 Averaging the data: SSMIS_UPP_Average

The `SSMIS_UPP_Average` routine (component 4), contained in the `SSMISMod_UPP_Average` module carries out the averaging of the brightness temperatures, passed to it from `ORBIT_DATA`.

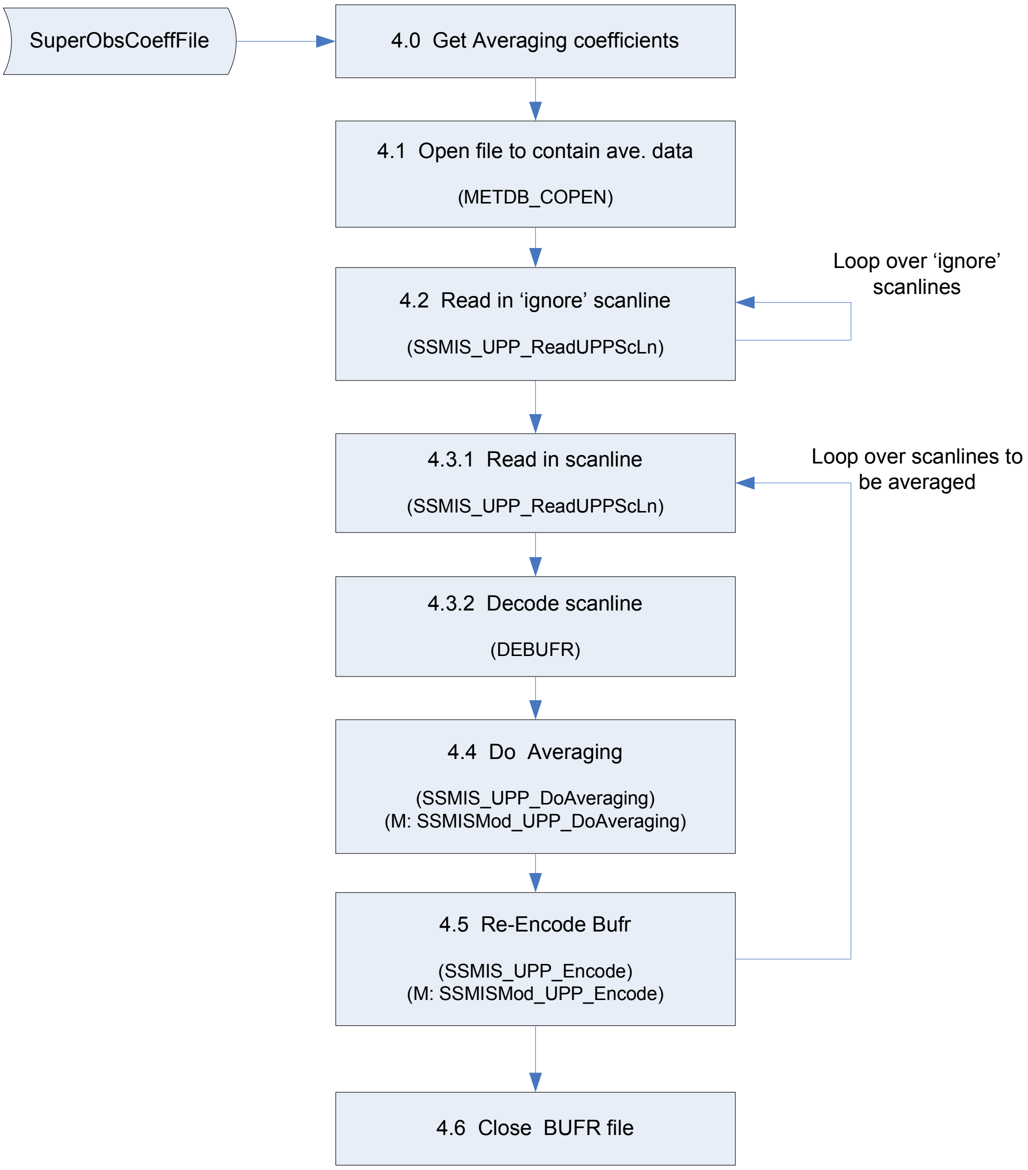


Figure 3: Flowchart showing the stages involved when calling the `SSMIS_UPP_Average`, subroutine. The subroutines called and the containing modules (denoted by M) are included in brackets for clarity. The numbering scheme directly refers to that within the code.

The averaging coefficients are initially read in (from `superob_coefficients.dat`, the name of which is specified by the `SuperObsCoeffFile` variable, declared in `SSMISMod_UPP`), as described above, and the file that is to contain the averaged data ("`UPP_BUFR_FILE_AV`"), is opened.

In a loop over all scan lines in the orbit the UPP data are read in from the BUFR file again, and the average brightness temperature values are then calculated using the algorithm described in Section 4 of the *Scientific Description.* This is achieved by calling the `SSMIS_UPP_DoAveraging` routine (component 4.4), contained in the `SSMISMod_UPP_DoAveraging` module. The averaged data are copied to the `NEW_UPP_VALUES_AV` array according to the specifications of the '`UPP_ToBeAveraged`' namelist, and passed back to the SSMIS_UPP_Average routine. (By default, all channels are averaged and the modified rain and surface flags are written out. However, the user may select just a few channels to be averaged, or retain the original rain or surface flags as required. See section 2.1.2.1).

Finally, the UPP data (in NEW_UPP_VALUES_AV) are encoded as a BUFR message and written to an external BUFR file. This is achieved by calling the SSMIS_UPP_Encode subroutine (component 4.5) (based on the SSMIS PP, SSMIS_PP_Encode.f90 subroutine) and is contained in the SSMISMod_UPP_Encode module.

When carrying out the averaging of the data, it is necessary to neglect a number of scan lines at the beginning and end of the orbit data. Therefore, a number of scan lines, (specified by the variable Ignore_scans_av, declared in SSMISMod_UPP) are read in (by calling SSMIS_UPP_ReadUPPScLn) but they are not decoded.

These functions are summarised in the flowchart shown in Figure 3.

2.1.7 Superobbing/Averaging Coefficients: superob_coefficients.dat

The averaging coefficients are read in by the routine SSMIS_UPP_Average prior to conducting the 'averaging' of the data. The data file read in contains the averaging coefficients for the 200 neighbouring points, (specified by the variable Domain_size, declared in SSMISMod_UPP) of each field of view in the scan line. For each coefficient, the scan position offset, scan line offset, latitude (not required), longitude (not required) and weighting are specified.

As for the SSMIS PP, the superobbing coefficients have been generated for a range of spatial averaging scales. The coefficients available are listed in Table 1 below:

Filename	**Description**
superob_coeffs_sigma1.dat	NO averaging. Weight for central fov =1
superob_coeffs_sigma25.dat	σ = 25km, FWHM=58.8km
superob_coeffs_sigma50.dat	σ = 50km, FWHM=117.7km
superob_coeffs_sigma75.dat	σ = 75km, FWHM=176.6km
superob_coeffs_sigma100.dat	σ = 100km, FWHM=235.4km
superob_coeffs_sigma150.dat	σ = 150km, FWHM=353.2km

Table 1. Averaging coefficient files and corresponding averaging scales for the SSMIS UPP pre-processor.

The averaging scale is set to σ = 50km (FWHM = 117.7km) (from superob_coeffs_sigma50.dat) by default, as this is used operationally by the UK Met Office global model. However, the averaging scale to be used will depend upon the application, for example, assimilation trials with the global model have used averaging scales given by σ = 100km (FWHM=235.4km). The desired averaging scale file can be selected either by copying the required coefficient file to the filename superob_coefficients.dat, or by editing the Makefile, as described in 3.4.

The format of the file is 60 blocks of data, in free format (space separated); with each block containing the following:

```
Fov #
Scan line offset, scan position, latitude (not required), longitude
(not required), weight (not normalised)
[repeated 200 times for each field of view]
```

For example:

```
1
0 0 44.78000 -58.27000 1.00000
-3 1 44.76000 -58.40000 0.97819
1 0 44.89000 -58.29000 0.97004
-1 0 44.67000 -58.24000 0.96943
-2 1 44.87000 -58.42000 0.95311
-4 1 44.65000 -58.38000 0.94469
```

The weights are then normalised within the code.

2.1.8 Output BUFR files: UPP_BUFR_FILE_AV.BUFR

The main output file containing the averaged data (the name of which is declared by the `Bufr_av_filename` variable in the module `SSMISMod_UPP` as “`UPP_BUFR_FILE_AV`”), takes the same format as the raw input BUFR file. The BUFR sequences are fully described by the BUFR sequence descriptor 51737. The Message Subtype is set to 7, indicating the output file contains UPP averaged data.

3 Installing and Using the SSMIS UPP Pre-processor

3.1 The directory structure

The SSMIS UPP averaging module is currently designed to operate using the directory structure of the SSMIS pre-processor, outlined in the *PP Technical description (p.17)*, with the UPP averaging module and namelist files being added into the structure. The revised directory structure is shown in Figure 4. The additional SSMIS UPP directories and files are highlighted in red.

Alternatively the UPP pre-processor may be run as a standalone package, in which case the directories specific to the *PP* package are absent.

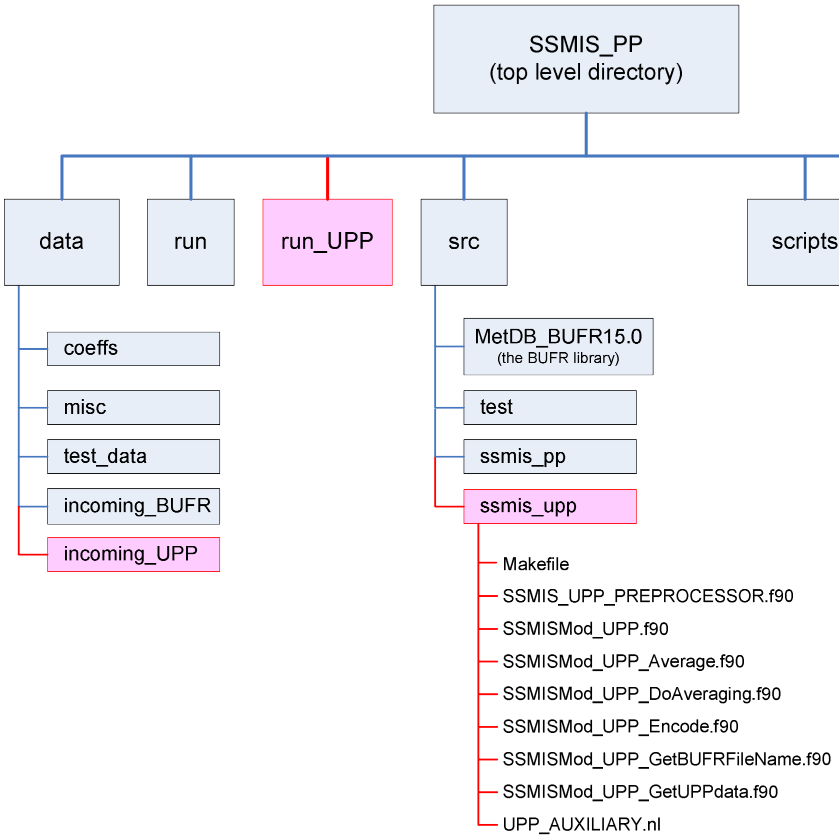


Figure 4: Directory structure of the SSMIS pre-processor, with additional SSMIS UPP pre-processor directories and files shown in red.

A brief description of the content of the additional directories for the SSMIS UPP pre-processor is given below:

`/SSMIS_PP/data/incoming_UPP` is intended for use as a repository for incoming (UPP) BUFR file

`/SSMIS_PP/run` is used by the SSMIS UPP pre-processor during execution

`/SSMIS_PP/src/ssmis_upp` contains the Makefile, source, auxiliary namelist and module files for the SSMIS UPP pre-processor

`/SSMIS_PP/archive_upp` is currently empty but can be used to archive pre-processed file, and/or input BUFR files

3.2 Unpacking the software

The UPP averaging module is supplied as a gzipped tar file. To allow installation as a standalone package, the tar file includes the BUFR library, `MetDB_BUFR15.0`, and the various `superob_coeffs_sigma*.dat` files. If you are adding the package to an existing SSMIS_PP installation, save the tar file to the directory *above* SSMIS_PP. Otherwise, save it to a convenient destination directory. Then unpack it with the command:

```
tar -xvzf ssmis_upp.tgz
```

or if your system does not support the "z" option, use the command:

```
gunzip -c ssmis_upp.tgz | tar -xvf -
```

The following directory structure will be created:

```
SSMIS_PP
    data
        coeffs
    src
        MetDB_BUFR15.0
        ssmis_upp
```

3.3 Building the library of BUFR routines

The pre-processor makes extensive use of a number of BUFR decode/encode routines. These routines are pre-compiled as a library that is linked during the compilation of the pre-processor. It is necessary therefore to compile the object library of BUFR routines as a first step. This is an identical step to that required by the SSMIS pre-processor and is also documented in the *PP Technical description*. This step is only necessary if not already running the SSMIS pre-processor. Various Makefiles are provided. For Linux, ifort and gfortran are recommended, e.g.

```
/SSMIS_PP/src/MetDB_BUFR15.0/Makefile_BUFRrelease_linux_ifort
```

The user may need to change the compiler flags and compiler specification to suit their environment. The library is made using the command:

```
make -f Makefile_BUFRrelease_linux_ifort
```

Further notes on the use of the BUFR routines can be found in the text file:

```
/SSMIS_PP/src/MetDB_BUFR15.0/BUFR_README.txt
```

The make should result in the generation of file:

```
libbufr.a
```

3.4 Building and compiling the SSMIS UPP averaging module

A script (`Makefile`, see Appendix 4) for building the pre-processor is given at:

```
/SSMIS_PP/src/ssmis_upp/Makefile
```

In this script, the superobbing coefficients file is specified according to the specific spatial averaging scale desired, *e.g.*

```
super_obcoefficients = $(COEFFS)/superob_coeffs_sigma50.dat
```

and may have to be edited by the user as required.

The Fortran compiler specified by the variable '`FF`' and the Fortran compiler flags specified by the variable `FFLAGS` may have to be changed according to the compiler selected by the user. (Use the same compiler as was used for the BUFR library).

Running (`make`) the `Makefile` script should compile the pre-processor main program and modules, link them to the library, and produce the executable:

```
/SSMIS_PP/src/ssmis_upp/SSMIS_PREPROCESSOR.exe
```

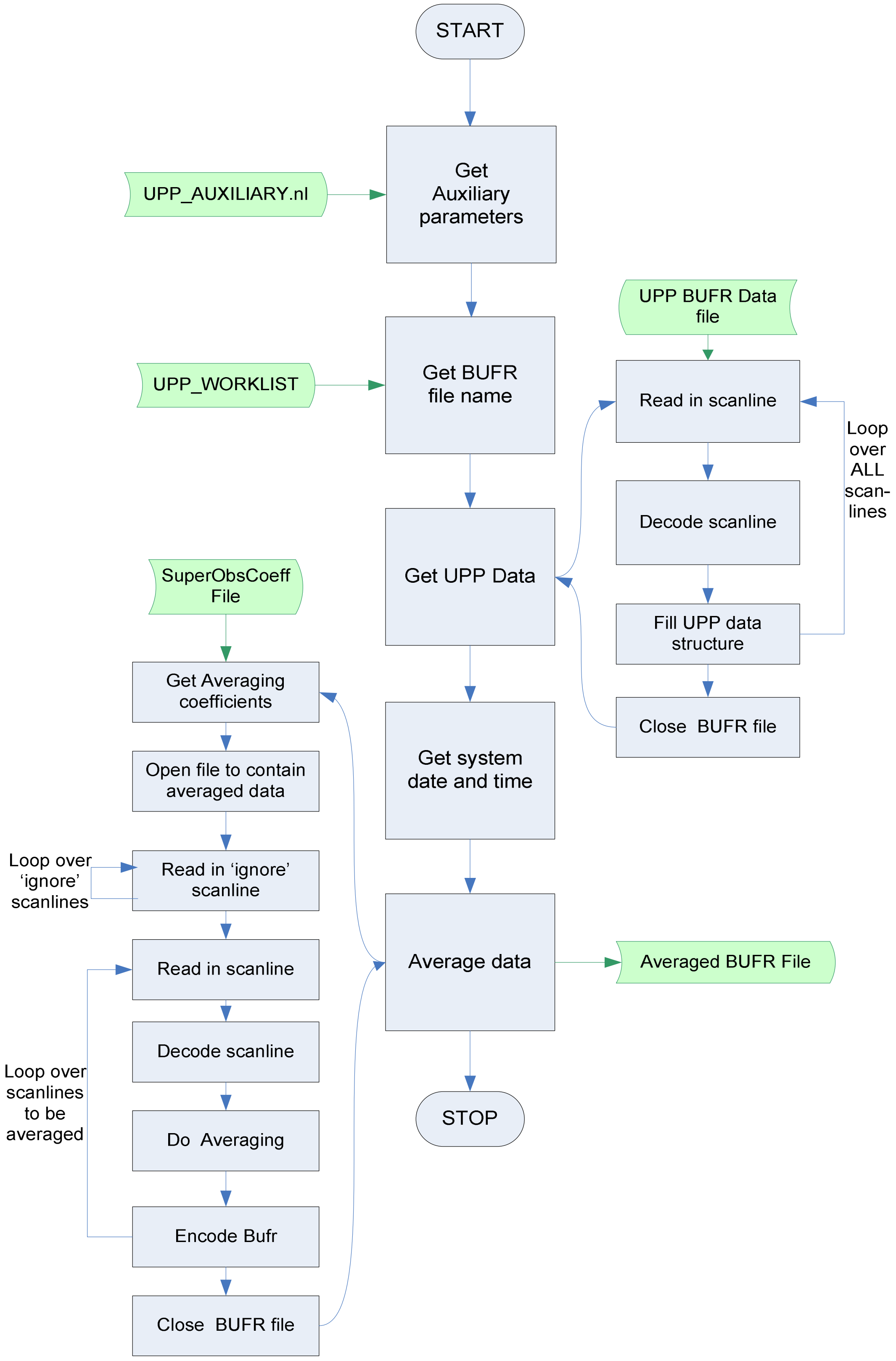
Then, installing this script (`make install`) should also copy the executable, the auxiliary namelist file (`UPP_AUXILIARY.nl`), the superobbing coefficients (superob_coefficients.dat) and the coefficient tables (TABLEB and TABLED) into the /SSMIS_PP/run_UPP directory.

Note: If the user wishes to alter the parameters in the UPP_AUXILIARY.nl file, the code does not need to be re-built, but the altered file does need to be copied into the run_UPP directory.

3.5 Running the pre-processor routinely

In order to run the pre-processor routinely it is necessary to prepare the UPP_WORKLIST with the name of the incoming BUFR file, as it is received. At the Met Office this is done using a script, controlled by cron to run every 15 minutes, which scans an incoming directory for the existence of the BUFR files. If any are present, the filenames are copied into the file UPP_WORKLIST, copied to the run_upp directory and the SSMIS UPP pre-processor is run. A copy of this script is included in Appendix 5.

Appendix 1 Flow Chart of the SSMIS UPP pre-processor



Appendix 2 Calling Tree for the SSMIS UPP pre-processor

```
SSMIS_UPP_PREPROCESSOR                                  !main program
        SSMIS_UPP_GetAux                                (0)
        SSMIS_UPP_GetBufrFileName                       (1)
        SSMIS_UPP_GetUPPdata                            (2)
                                                        !Begin loop all ScLn
                SSMIS_UPP_ReadUPPScLn                     (2.1.1)
                        BUFRREAD_LAS
                DEBUFR                                    (2.1.2)
                DESFXY                                    (2.13)
                                                        !End loop all ScLn
        SSMIS_UPP_Datime                                (3)
                DATE_AND_TIME
        SSMIS_UPP_Average                               (4)
                METDB_COPEN                               (4.1)
                                                        !Begin loop ‘ignore’ ScLn
                SSMIS_UPP_ReadUPPScLn                     (4.2.1)
                        BUFRREAD_LAS
                                                        !End lp ‘ignore’ ScLn
                                                        !Begin loop ‘averaging’ ScLn
                SSMIS_UPP_ReadUPPScLn                     (4.3.1)
                        BUFRREAD_LAS
                DEBUFR                                    (4.3.2)
                SSMIS_UPP_DoAveraging                     (4.4)
                SSMIS_UPP_Encode                          (4.5)
                        ENVBUFV2
                        METDB_CWRITE
                                                        !End loop ‘averaging’ ScLn
                METDB_CCLOSE                              (4.6)
```

Unless referred to in the main text in section 2, the routines are found in the BUFR library, MetDB_DB15.0. The routines’ numbering scheme refers to that included within the code.

(Abbreviation: Scan lines, ScLn)

Appendix 3 The UPP_AUXILIARY.nl file

```
!*************************************************************************************
!  UPP_AUXILIARY.nl file
!*************************************************************************************
! Description:
!  Namelist file to specify control settings for the SSMIS UPP (averaging)
!  pre-processor.
!
!  It is not compulsory to include any parameters within the namelists in this file
!  because all switches etc have appropriate default values assigned in the code.
!  The purpose of this file is to allow the user to change various settings without
!  needing to make code changes.
!
!  The namelists will be read in by subroutine SSMIS_UPP_GetAux, defined in module
!  SSMISMod_UPP.
!
!  Namelist:  'UPP_ToBeAveraged'
!   Contains the following parameters:
!     ChannelsToBeAveraged  - This is the list of channels that are to be averaged.
!                           - It will default to average All channels.
!     SetAveragedRainFlags  - This enables the original rain flags to be kept if
!                             this is switched off.
!                           - Default is 'ON' / '.TRUE.'
!     SetSurfaceDiscrepancy - This enables the original surface types to be kept
!                             if this flag is switched off.
!                           - Default is 'ON' / '.TRUE.'
!
!
! Method:
!   Although it is not compulsory to include parameters within the namelist, the
!   namelist itself must be included.  Including the following code will produce
!   default results:
!
!   &UPP_ToBeAveraged
!   /
!
!   Alternatively use:
!
!   &UPP_ToBeAveraged
!   ChannelsToBeAveraged=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24
!   SetAveragedRainFlags = .TRUE.
!   SetSurfaceDiscrepancy= .TRUE.
!   /
!
!
! History:
!
! Version  Date       Modification history:
! -------  --------   ---------------------
! 1.0      12/11/10   New file.  Anna Booton
!-----------------------------------------------------------------------------

   &UPP_ToBeAveraged
   /
```

Appendix 4 Building the SSMIS UPP pre-processor

'`Makefile`' Script:

```
# Makefile for SSMIS_UPP
#
# 19/11/2009 Nigel Atkinson
##################################################################

#Compiler options

FF          = ifort
FFLAGS      = -i-static -g  -w90 -w95 -cm
#FF          = gfortran
#FFLAGS      = -g -O1

#Directories

TOP         = $(shell cd ../..; pwd)
RUN         = $(TOP)/run_UPP
COEFFS      = $(TOP)/data/coeffs
MDBBUFR     = $(TOP)/src/MetDB_BUFR15.0

#Data files to be linked in the run directory

TABLEB = $(MDBBUFR)/TABLEB
TABLED = $(MDBBUFR)/TABLED
CODEFIG = $(MDBBUFR)/CODEFIG
superob_coefficients = $(COEFFS)/superob_coeffs_sigma50.dat
AUXILIARYNL = UPP_AUXILIARY.nl

LIB         = $(MDBBUFR)/libbufr.a

BIN         = SSMIS_UPP_PREPROCESSOR.exe

OBJ         = \
      SSMISMod_UPP.o \
      SSMISMod_UPP_GetBUFRFileName.o \
      SSMISMod_UPP_GetUPPdata.o \
      SSMISMod_UPP_Datime.o \
      SSMISMod_UPP_DoAveraging.o \
      SSMISMod_UPP_Encode.o \
      SSMISMod_UPP_Average.o

#Pattern rule: each object file depends on its corresponding f90 file.
#The order of modules matters.
# $@ refers to the left side of the dependency line
# $< refers to the first item on the right side
# $^ refers to all items on the right side

%.o: %.f90
      $(FF) -c $(FFLAGS) -o $@ $<

%.exe: %.f90 $(OBJ) $(LIB)
      $(FF) $(FFLAGS) -o $@ $^
```

```
#Default goal:

bin: $(BIN)

clean:
      rm -f $(OBJ) $(BIN) *.mod

install:
      mkdir -p $(RUN)
      cp $(BIN) $(RUN)
      cp $(AUXILIARYNL) $(RUN)
      ln -sf $(TABLEB) $(RUN)
      ln -sf $(TABLED) $(RUN)
      ln -sf $(CODEFIG) $(RUN)
      ln -sf $(superob_coefficients) $(RUN)/superob_coefficients.dat
```

Appendix 5 Running the SSMIS UPP pre-processor

Below is a copy of the script used at the Met Office for routinely running the SSMIS UPP pre-processor. It is an example script and would need to be adapted by the user. The UPP files are processed, then transferred to the Met Office's observational database, MetDB.

```
#!/usr/bin/ksh

#---------------------------------------------------------------------------
#
# Script to run the SSMIS UPP PREPROCESSOR and optionally send output
# to MetDB. Run from cron.
#
# Assumes the SSMIS_UPP_PREPROCESSOR has been built and installed in the
# "run" directory (including links to coefficient files, BUFR tables, etc.)
#
# Incoming data are in directory "data/incoming_UPP".
#
# External script: transfer_to_metdb
#
# Version    Date    Author   Comments
# =======    ====    ======   ========
#    1.0  09/11/09   NCA
#
#---------------------------------------------------------------------------

TOP=$(dirname $0)/..     #Assume we are in a "scripts" directory

IN_DIR=$TOP/data/incoming_UPP
ARCHIVE_DIR=$TOP/archive_UPP
RUN_DIR=$TOP/run_UPP

mkdir -p $ARCHIVE_DIR

#Edit the following two lines as required:
#-----------------------------------------
sats_required="SA SB SC"       #SA=F16, SB=F17, SC=F18
metdb_required="N N N"
#-----------------------------------------

function log {
  echo $(date +%Y%m%d_%H%M) $* >>LOGFILE_UPP
}

cd $RUN_DIR

files=$(cd $IN_DIR; ls *_NS)       #ignore *.tmp files

#trim log file if this is the first file of the day

if [ "$files" != "" ]; then
  lastdate=$(tail -1 LOGFILE_UPP | cut -c1-8)
  now=$(date +%Y%m%d)
  if [ "$now" != "$lastdate" ]; then
    grep $lastdate LOGFILE_UPP >LOGFILE_UPP_tmp
    mv LOGFILE_UPP_tmp LOGFILE_UPP
  fi
fi

for file in $files; do

#See whether this file is required to be processed

  process=N
  metdb=N
  set $metdb_required               #copy $metdb_required => $1, $2, etc.
  for sat in $sats_required; do
    if [[ $file == *${sat}* ]]; then
      process=Y
      metdb=$1
    fi
    shift       #move $2 => $1
  done
  if [ $process = Y ]; then
    mv $IN_DIR/$file .
    echo $file >UPP_WORKLIST

#Work out the delay in minutes

    datetime=$(echo $file | cut -f 3-4 -d _)
    startyear=$(echo $datetime | cut -c2-3)
    startday=$(echo $datetime | cut -c4-6)
    starttime=$(echo $datetime | cut -c9-10):$(echo $datetime | cut -c11-12)
    startsecs=$(date +%s -d "20${startyear}0101 $starttime + \
              $(expr $startday - 1) days")
    nowsecs=$(date +%s)
    delay=$(expr $nowsecs - $startsecs)
    delay=$(expr $delay / 60)
    log "IN $file delay $delay"

#Run the pre-processor

    SSMIS_UPP_PREPROCESSOR.exe >SSMIS_UPP_PREPROCESSOR.out 2>&1

    status=$?
    if [ $status = 0 ] && [ -s UPP_BUFR_FILE_AV.BUFR ]; then
      if [ $metdb = Y ]; then

#Send output file to MetDB

        transfer_to_metdb UPP_BUFR_FILE_AV.BUFR SSMIS
        if [ $? = 0 ]; then
          log "OUT $file to MetDB"
        fi
      fi

#Archive the output file

      mv UPP_BUFR_FILE_AV.BUFR $ARCHIVE_DIR/${file}_AV.bufr
    else
      log "SSMIS_UPP_PREPROCESSOR.exe failed for $file"
    fi

#Archive the input file

    mv $file $ARCHIVE_DIR
  else
    mv $IN_DIR/$file $ARCHIVE_DIR     #no processing required
  fi

done

#Clean up the archive directory (remove files more than 1 day old)

cd $ARCHIVE_DIR
find . -type f -mtime +0 -maxdepth 0 -exec rm {} \;
```