A PREPROCESSOR FOR SSMIS RADIANCES

Technical Description

W. Bell July 2006

Version	Date	Author	Remarks	Reviewed	Approved
1.0	**06/04/06**	**W. Bell**	**Initial Version**	**N. Atkinson (21/04/06)**	**S. English (12/06/06)**
1.1	**25/07/06**	**W. Bell**	**Update following review by DF/PS**	**S. English (27/07/06)**	**B. Conway (27/07/06)**

TABLE OF CONTENTS

Scope

This document describes the overall design of the NWP SAF SSMIS pre-processor. The aims of this document are to:

- give the user a clear and simple overview of how the pre-processor is organised
- describe the operation of the main subroutines
- specify the formats and contents of the input and output files
- describe the approach to testing, both scientific and technical testing, and the results obtained
- provide a guide to enable users to build and run the code

The *Code Overview* section gives a brief account of the design of the code which, together with the flowchart and calling tree included as *Appendices A* and *B* respectively, should give the user a clear understanding of the organisation of the pre-processor.

A more detailed account of the most important subroutines is described in the section *Description of the Main Subroutines* This section supplements the in-code documentation. The section on *Testing* covers both scientific and software testing , although reference is made to the *Scientific Description* where the scientific testing is covered in more detail. The approach to software testing, involving both compiler and platform testing, is described here.

Guidance on installing and running the pre-processor is given in the final section, *Installing and Running the SSMIS Preprocessor.*

Related documentation

A Preprocessor for SSMIS Radiances Scientific Description, hereafter referred to as the *Scientific Description.*

Code Overview

Introduction

The main functions of the pre-processor, as described in the *Scientific Description*, are to:

- Remap the data to a common grid
- Average the data over selectable spatial scales
- Flag observations affected by solar intrusions
- Correct the measured brightness temperatures for reflector emission

The pre-processor takes as input four BUFR files per orbit, corresponding to the four SSMIS instrument subtypes: lower atmospheric sounder data (LAS); imager data (IMA); environmental data (ENV) and upper atmospheric sounder data (UAS). Details of the channels associated with these subtypes are given in Table 1 of the *Scientific Description*. The pre-processor generates as output:

- Remapped data, as a BUFR file
- Remapped and averaged data, as a BUFR file
- Auxilliary ASCII files to assist in diagnosing the performance of the pre-processor

A flowchart for the SSMIS pre-processor is given in *Appendix A*. The calling tree is given in *Appendix B*.

An important part of the code involves setting up working arrays containing the SSMIS radiance data and associated auxiliary data. This data is stored in *f90* derived structures. The brightness temperature data is stored in three dimensional (3D) arrays within these structures. These 3D arrays are indexed by scan line (*ie along track* scan line index), scan position (*ie 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, and compiling the pre-processor are given in the section *Installing and Running the Pre-processor.*

The data for the IMA, ENV and UAS datasets are read in first for a whole orbit. In order to perform the solar intrusion flagging and reflector emission correction additional information has to be calculated (*eg* local solar zenith and azimuth angles, constructed reflector temperature). The resulting data is stored in the IMA data structure (`IMA_DATA`) for later use. In addition external data, required for the corrections and flagging, is read in (*eg* the solar intrusion map, coefficients required for the correction of reflector emission *etc*).

The LAS data is then read, one scan line (*ie* one BUFR message) at a time, and the other channels are remapped to the LAS grid following the algorithm described in Section 3 of the *Scientific Description*. This data is then written out to a BUFR file (remapped but not averaged) and is also stored in a new data structure (`ORBIT_DATA`) for subsequent processing.

If averaging has been selected then the LAS data are read in 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 is re-encoded as a BUFR message and output to a new BUFR file (remapped and averaged). 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.*

Input Data/ Working arrays

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

- **Instrument data** comprising the brightness temperature data from the four SSMIS instrument subtypes (LAS, ENV, IMA and UAS data) together with related data (latitudes, longitudes, time of the observation, surface and rain flags, onboard calibration data *etc*). All of this data is present in the SSMIS BUFR data files.
- **Coefficient data** required by the correction, remapping and averaging routines. This is read in from ASCII files.

The instrument data is available initially as external binary BUFR files. There is normally one BUFR file per orbit per instrument subtype (*ie* there are four files per orbit). As an example, the raw BUFR filenames take the form:

`NPR_TDUB.SA_D06111_S0430_E0608_B1293233_NS:` UAS data
`NPR_TDLB.SA_D06111_S0430_E0608_B1293233_NS:` LAS data
`NPR_TDEB.SA_D06111_S0430_E0608_B1293233_NS:` ENV data
`NPR_TDIB.SA_D06111_S0430_E0608_B1293233_NS:` IMA data

The sixth field (*eg* here `B1293233`) encodes the orbit number, in this case the orbit commencing revolution 12932, and ending with the beginning of revolution 12933. For example, the next orbit will have `B1293334 in` the sixth field.

These filenames are made available to the pre-processor via a text file (`WORKLIST`). The file `WORKLIST` can be prepared by a Unix shell script which scans an incoming directory for new BUFR files. An example shell script to do this is included in *Appendix C*. The text file containing these names is read in by the routine `SSMIS_PP_GetBUFRfilenames`. The first stage in processing involves reading all brightness temperature data, and important auxiliary data, into arrays within derived data structures (`IMA_DATA, ENV_DATA` *etc*). This input is carried out in the routines:

- `SSMIS_PP_GetIMAdata`
- `SSMIS_PP_GetUASdata`
- `SSMIS_PP_GetENVdata`

The brightness temperature data for the IMA, ENV and UAS scans are read into derived type data structures and all relevant coefficient files are read in *before* the LAS data is read in using the routine:

- `SSMIS_PP_GetLASdata`

These routines make use of lower level BUFR read and decode routines. The coefficient files are read in at various points in the code. The format and content of these files are described in more detail in the Section on *Input files*. The coefficient files are listed in Table 1 below:

Coefficient filename	Description
`superob_coefficients.dat`	Coefficients for the averaging of the data using 200 neighbouring points for each LAS scan position
`SSMIS_intrusion_map.dat`	Solar intrusion map for detection of observations affected by solar intrusions
`ima_interpolation_coeffs.dat`	Coefficients used for the remapping of the data from the IMA grid to the LAS grid
`env_interpolation_coeffs.dat`	Coefficients used for the remapping of the data from the ENV grid to the LAS grid
`uas_interpolation_coeffs.dat`	Coefficients used for the remapping of the data from the ENV grid to the LAS grid
`T_ARM_TEMPLATE.DAT`	Data required for the synthesis of the reflector temperature
`TB_CORRECTIONCOEFFS.DAT`	Spillover correction factors and effective emissivities
`T_CORRECTION_FACTOR.DAT`	Parameters used in the generation of the reflector temperature

Table 1: Input coefficient files required by the SSMIS pre-processor

Output diagnostic files

A number of files are output which can be used for diagnostics and monitoring. The most important of these are dumps of the `IMA_DATA` structure which contains information on the calibration of each channel, the time evolution of the reflector arm temperature as well as the constructed reflector temperature (see Section 6 of the *Scientific Description*). The format and content of these files is described more fully in the section on *Output files*, but they are listed below in Table 2:

Diagnostic filename	Description
UAS_DIAGNOSTICS_MAP.DAT	Output of raw UAS data for checking remapping performance
UAS_DIAGNOSTICS_REMAP.DAT	Output of remapped UAS data for checking remapping performance
ENV_DIAGNOSTICS_MAP.DAT	Output of raw ENV data for checking remapping perfromance
ENV_DIAGNOSTICS_REMAP.DAT	Output of remapped ENV data for checking remapping performance
IMA_DIAGNOSTICS_AUX.DAT	Output of IMA calibration and auxiliary data
IMA_DIAGNOSTICS_MAP.DAT	Output of raw IMA data for checking remapping perfromance
IMA_DIAGNOSTICS_REMAP.DAT	Output of remapped IMA data for checking remapping performance

Table 2. Output files generated by the SSMIS pre-processor.

Intrusion mapping

A series of routines are called from the `SSMIS_PP_GenRemapLASBUFR` routine. After the IMA data is read in (`SSMIS_PP_GetIMA`) `SSMIS_PP_CalcSolZenAndAz` is called and the solar zenith and azimuth angles are calculated for each scan line. The solar zenith and azimuth angles are computed for the central ground footprint location (at the *ground*) for each scan line rather than at the location of the spacecraft itself. This information is used in conjunction with a solar intrusion map to determine whether a given scan line is affected by solar intrusions into the warm calibration load. The solar intrusion map takes the form of a 360×180 array (spanning azimuth angles of 0°-360°, and zenith angles of 0°-180°) of 0's and 1's which is read in from an external ASCII file (`SSMIS_intrusion_map.dat`) as a single column. The routine `SSMIS_PP_DetectIntrusions` determines whether the computed local solar zenith and azimuth angles for each scan line are within an intrusion affected region (*ie* relevant map values at zenith and azimuth indices = 1) or not (map value = 0).

Remapping data

Once the IMA, ENV and UAS data have been read into the relevant data structures (`IMA_DATA` *etc*) for a complete orbit, then the LAS data is read in one scan line at a time. The top level calling routine which controls this section of the code is `SSMIS_PP_ProcessLAS`. At the start of each orbit a fixed number of LAS scan lines are read in and ignored (using `SSMIS_PP_GetLASAndIgnore`) to allow for the remapping routines to work. The remapping routines effectively interpolate brightness temperatures from neighbouring fields of view and the offset in the scan lines for some channels means that at the start and end of the LAS scans this cannot be done. For each scan line the routine `SSMIS_PP_RemapToLAS` is called which uses pre-computed remapping coefficients read in from an external ASCII file (`ima_interpolation_coeffs.dat` *etc*). The brightness temperatures for the IMA, ENV and UAS channels are remapped to the LAS footprints as described in section 3 of the *Scientific Description*. The remapped data for each scan line is (1) encoded as BUFR and output to a BUFR file and (2) stored in the data structure `ORBIT_DATA` for subsequent averaging, if averaging is requested. The corrections for the reflector emission effects are carried out in a call to the routine `SSMIS_PP_CorrectTB` (from `SSMIS_PP_RemaptoLAS`). This uses spillover correction coefficients and effective emissivities from an external ASCII file (`TB_CORRECTIONCOEFFS.DAT`).

Averaging data

If the logical variable (`SSMIS_DoAveraging`) is set to *true* in the routine `SSMIS_PP_ProcessLAS` then the remapped data stored in `ORBIT_DATA` is averaged. The averaging coefficients are read in from an external file (`superob_coefficients.dat`) and stored in the structure `coefficients`. The brightness temperature data for the entire orbit is averaged using the algorithm described in Section 4 of the *Scientific Description*. The implementation involves reading in the LAS data from the BUFR file again, scan line-by-scan line, and copying the averaged brightness temperatures (from the `ORBIT_DATA` structure) to the relevant part of the decoded BUFR message (*ie* to the appropriate slot in the *values* array). The averaged data is output to a BUFR file in exactly the same format as the input BUFR files, except all 24 brightness temperatures are present, and the rain and surface flags have been modified as a result of the remapping and averaging.

Correcting for Reflector Emission

The effect and the overall correction strategy is described in detail in Section 6 of the *Scientific Description.* There are two main steps involved in correcting the measured brightness temperatures for the effects of thermal emission from the main reflector.

Firstly the main reflector temperature is constructed from the measured temperature data from the reflector arm. This is carried out soon after the raw data is read in from the IMA BUFR file, in the routine `SSMIS_PP_GenTant`, called from `SSMIS_PP_GenRemapLASBufr.` The constructed reflector temperature is stored in the `IMA_DATA` structure (in `IMA_DATA % T_ANT`). Secondly, the correction to the brightness temperatures, given by Equation 7 in Section 6 of the *Scientific Description*, is carried out in the routine `SSMIS_PP_CorrectTB`, which is called during the processing of each scan line in the routine `SSMIS_PP_RemapToLAS`.

Description of main subroutines

Main Program and top level subroutine

The main program `SSMIS_PREPROCESSOR` has no input or output arguments. The main program has two functions:

- setting up an array of four filenames corresponding to the four BUFR input files. This is done by calling the routine `SSMIS_PP_GetBufrfiles` which has an character array output argument `bufr_filenenames`.

- calling the subroutine `SSMIS_PP_GenRemapLASBUFR` where all subsequent processing is done.

Reading in the IMA data from the BUFR file: SSMIS_PP_GetIMA

The low level BUFR routines which read the BUFR data are common to the parts of the code concerned with reading in the: LAS, UAS, IMA and ENV data (although they have been renamed separately) so only those routines concerned with the IMA data are described here.

Firstly BUFR messages are read in turn from the BUFR file by a call to the subroutine

BUFRREAD_IMA. Each BUFR message contains data from a single scan line. In the case of the IMA data this corresponds to 180 fields of view, plus associated auxiliary data. The subroutine DEBUFR is then used to decode the BUFR message and the resulting data is returned in *descriptor* (containing code numbers associated with descriptions of the data) and *values* arrays. The following data from the raw IMA data file is stored in the IMA_DATA structure for subsequent processing:

- Brightness temperatures for all subtype channels (for IMA data this is 8-11,17 and 18)
- Warm load temperatures from the three temperature sensors in the warm calibration load
- The warm and cold counts for each calibration (once per scan line) for each of the 24 channels
- Multiplexed temperature data streams, from various monitoring points on the instrument
- The temperature of the arm supporting the main reflector, later used in the construction of the reflector temperature
- Surface code
- Rain flag associated with the data
- Latitude and longitude of each observation
- Scan line and scan position for each observation
- Year, month, day, hour, minute and second for each observation

Processing the LAS data: SSMIS_PP_ProcessLAS

The main remapping and correction/flagging of the data is carried out in the routine SSMIS_PP_ProcessLAS.

This routine firstly reads in a fixed number of scans (normally 15) from the LAS BUFR file and ignores these. The reason for this is that the remapping is effectively an interpolation and the offset between different scan lines for the four data types means that this interpolation is not possible for the 15 scans lines at the beginning and end of an orbit.

The remapping coefficients are read in from external ASCII files as are the correction coefficients, prior to the commencement of the loop over all LAS scan lines. Within the scan line loop the BUFR message is read from the LAS BUFR file using the routine (SSMIS_PP_GetLAS). The BUFR messages are decoded to produce *values* and *descriptor* arrays, The routine SSMIS_PP_RemaptoLAS is called within the scan line loop. In SSMIS_PP_RemaptoLAS the remapping to the LAS grid is carried out. Surface codes and rain flags are also remapped. The reflector emission correction is carried out by a call to SSMIS_PP_CorrectTB.

Finally the modified LAS data, complete with remapped data for all 24 channels and remapped surface and rain flags are encoded and written to a BUFR file.

Averaging the Data: SSMIS_PP_Average

The remapped data for the complete orbit is stored in the routine SSMIS_PP_ProcessLAS. The data is contained in the *f90* derived structure ORBIT_DATA, which contains only the following information:

Brightness temperatures
Rain flag

Surface flags
Scan line number

If the logical switch `SSMIS_DoAveraging` is set to true then the routine `SSMIS_PP_Average` is called and `ORBIT_DATA` is passed to it. In `SSMIS_PP_Average` the averaging coefficients are read in from the external ASCII file and then, in a loop over all scan lines in the orbit (neglecting a fixed number of scans at the beginning and end of each orbit, for reasons described above) the LAS data is read in from the BUFR file again, the relevant scan lines from `ORBIT_DATA` are averaged using the algorithm described in Section 4 of the *Scientific Description*, and the averaged data is copied across to the LAS scan line data. Finally the LAS data is encoded as a BUFR message and written to an external BUFR file.

Input files

Superobbing/Averaging Coefficients : superob_coefficients.dat

The averaging coefficients are read in by the routine `SSMIS_PP_Average` prior to the loop over all scan lines in the `ORBIT_DATA` array. The superobbing coefficients have been generated for a range of spatial averaging scales. The coefficients available are listed in Table 3 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 3. Averaging coefficient files and corresponding averaging scales for the SSMIS Pre-processor

The averaging scale to be used will depend upon the application, for example, assimilation trials with the UK Met Office global model have used averaging scales given by σ = 100km (FWHM=235.4km). The file can be selected either by copying the required coefficient file to the filename `superob_coefficients.dat`, or by soft linking the relevant coefficient file to this filename, *eg* :

```
ln –s superob_coeffs_sigma100.dat superob_coefficients.dat
```

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.

SSMIS Intrusion Map : SSMIS_Intrusion_map.dat

The solar intrusion map is read in by the routine `SSMIS_PP_DetectSolarIntrusions`. The map takes the form of an ascii file containing a list of (180×360) reals (0 or 1) as a single column, the data is read into `SSMIS_PP_DetectSolarIntrusions` by the following lines of code:

```
Do j=1,360
  Do i=1,180
    read(intrusion_map_number,*) map_real(i,j)
    map_integer(i,j) = int( map_real(i,j) )
  End Do
End Do
```

Remapping coefficients for the IMA,ENV and UAS data:

The remapping coefficients are read in by the routine, `SSMIS_PP_GetRemapCoeffs` which is called by `SSMIS_PP_ProcessLAS`. The remapping coefficient files for the IMA,ENV and UAS data share the same space separated free format, which is :

```
Relative scan line_1, scan position_1, weight_1, Relative scan line_2,
scan position_2, weight_2, Relative scan line_3, scan position_3,
weight_3, Relative scan line_4, scan position_4, weight_4.
```

For example:

-2 4 0.56881 -3 5 0.18655 -1 3 0.15601 0 2 0.08864
-2 7 0.44916 -1 6 0.26941 -3 8 0.15368 0 5 0.12774
-2 10 0.36231 -1 9 0.35805 0 8 0.14764 -3 11 0.13200
-1 12 0.39462 -2 13 0.32829 0 11 0.14692 -3 14 0.13016
-1 15 0.48516 -2 16 0.23066 0 14 0.17162 -3 17 0.11256

This is repeated 60 times corresponding to the 60 LAS fields of view.

Arm template temperature: T_ARM_TEMPLATE.DAT

The construction of the reflector face temperature requires the computation of a *lagged* derivative of the reflector arm temperature data set as described in section 6 of the *Scientific Description*, and is generated for each orbit as received. At the beginning of the orbit the arm temperature data (and its derivative) for the end of the previous orbit is required to generate the lagged derivative for the new orbit. The previous arm temperature data (from the end of the previous orbit) is not available to the code. The solution to this has been to generate a template of arm temperatures and associated time derivatives. The new orbit data is matched to this and data from the template is used for the period immediately before the start of the new orbit. This template data is stored as an external ASCII file which is read in by the routine `SSMIS_PP_GenTant`. It is formatted as follows (space separated free format):

```
Number of data points
T_ARM temperature, derivative of T_ARM wrt time
```

For example:

```
13000
  2.7937400e+02  -2.5284800e-02
  2.7934872e+02  -2.5308800e-02
  2.7932366e+02  -2.5332000e-02
  2.7929868e+02  -2.5347600e-02
  2.7927370e+02  -2.5349200e-02
  2.7924874e+02  -2.5337600e-02
…. etc
```

Brightness temperature correction coefficients :TB_CORRECTIONCOEFFS.DAT

The brightness temperature correction coefficients are read in by the routine `SSMIS_PP_GetTBCorrCoeffs`, which is called by `SSMIS_PP_ProcessLAS`. The brightness temperature correction coefficients file contains the spillover correction ($K_{CH\#}$) for each channel together with the effective emissivity ($\varepsilon^{eff}_{CH\#}$) for each channel:

K_{CH1} ε^{eff}_{CH1}
K_{CH2} ε^{eff}_{CH2}
K_{CH3} ε^{eff}_{CH3}

The file is space separated and free formatted. For example:

```
0.980  0.01
0.985  0.01
0.988  0.01
0.989  0.01
0.985  0.01
... etc
```

Reflector temperature correction factors: T_CORRECTION_FACTOR.DAT

This file is read in by the routine `SSMIS_PP_GenTant`. The file contains two coefficients used in constructing the reflector face temperature from the measured reflector arm temperature: the value of the scale factor used to scale the correction to the measured arm temperature, and the width of the lag applied to this correction. These are described in Section 6 of the *Scientific Description* (see c_2 and σ in Equation 9 respectively).

Output files

Output BUFR files: LAS_BUFR_FILE.BUFR and LAS_BUFR_FILE_AV.BUFR

The main output files are the remapped BUFR data (`LAS_BUFR_FILE.BUFR`) and the remapped and averaged data (`LAS_BUFR_FILE_AV.BUFR`). These files take the same format as the raw input BUFR files and the BUFR sequences are fully described by the BUFR sequence descriptor 51737.

Diagnostic output files with remapping data

The files `UAS_DIAGNOSTICS_MAP.DAT` and `UAS_DIAGNOSTICS_REMAP.DAT`

Contain data from small sections of the orbit that allow the user to regenerate remapping coefficients and to check performance of the remapping routines.

The file `UAS_DIAGNOSTICS_MAP.DAT` contains the original unmapped UAS data for UAS channels 19-23 and is output by the routine `SSMIS_PP_OutputDiagnosticsUAS`. The format of the data is (7F9.3):

```
latitude, longitude, TB_ch19,TB_ch20,TB_ch21,TB_ch22, TB_ch23
```

The file `UAS_DIAGNOSTICS_REMAP.DAT` contains the following data:

```
scan line, scan position, latitude and longitude
```

in the format (I4,1X,I3,1X,F8.3,1X,F8.3) for scan lines 970 to 1030. This data can be used in conjunction with the LAS data to compute remapping coefficients. Similar files exist for IMA and ENV data.

IMA data auxiliary file

The IMA data auxiliary file `IMA_DIAGNOSTICS_AUX.DAT` is perhaps the most important of the output diagnostics files and contains important information on the calibration of the instrument, the intrusions flags computed and the reflector emission corrections applied. The file format is:

(F8.3,1X,F8.3,1X,I4,1X,I2,1X,I2,1X,I2,1X,I2,1X,F5.2,1X,F6.2,1X,F6.2,1X,7(F8.3,1X),
48(I6,1X),1X,I1,1X,I2,1X,I4,1X,I4,1X,4(F8.3,1X))

And the fields are described in Table 4 below:

FIELD #	FORMAT	DESCRIPTION
1	F8.3	latitude
2	F8.3	longitude
3	I4	year
4	I2	month
5	I2	day
6	I2	hour
7	I2	minute
8	I2	second
9	F5.2	Solar zenith (gnd footprint)
10	F6.2	Solar azimuth (gnd footprint)
11 - 13	3F8.3	Warm load temperature /K
14 - 17	4F8.3	Multiplexed data records 1-4
18 - 41	24I6	Warm load counts
42 - 65	24I6	Cold load counts
66	I1	Surface code
67	I2	Rain flag
68	I4	Scan line
69	I4	Nearest scan line
70	F8.3	Measured arm temp /K (T_ARM)
71	F8.3	Measured arm temp (interpolated)
72	F8.3	Reflector correction / K
73	F8.3	Computed reflector temperature

Table 4: Contents and format of the `IMA_DIAGNOSTICS_AUX.DAT` file.

Testing

Two sets of challenges were faced in developing and testing the SSMIS pre-processor. Firstly the input radiances were affected by two significant and complex instrumental biases. An important function of the pre-processor is to correct for these biases to a level where the radiances might be expected to deliver significant benefit when assimilated in an NWP forecast model. Defining precisely the error characteristics of a radiance dataset which will achieve this goal is not straightforward. Nevertheless, experience gained in the processing of similar radiance datasets (eg from AMSU on ATOVS) permits us to define criteria for the testing of the output from the pre-processor. Ideally, the validation of the output dataset should involve the comparison of the pre-processed radiances with an independent dataset with well established metrological traceability. Unfortunately this type of idealised validation is often not possible for satellite measured radiances. Test criteria are therefore specified in terms of differences between the output radiances and modelled radiances. The modelled radiances are based on radiative transfer modelling using short range (T+6 hours) forecast fields from a global NWP model. Secondly, there are technical challenges associated with developing a pre-processor which is sufficiently efficient that significant delays are not introduced into the radiance processing system, and with ensuring cross compiler and platform compatibility.

Testing was therefore carried out at two levels: *scientific testing*, aimed at establishing whether the corrections currently applied to the SSMIS radiances in the pre-processor are of sufficient quality to provide positive impact on forecast quality when the data is assimilated in a global NWP system, and secondly *technical testing*, aimed at verifying that the code will produce consistent results when compiled using different compilers (*compiler testing*) and when compiled and run on different platforms (*platform testing*) and meet the timing requirement set out in the *Product Specification*. These tests are described below.

Scientific Testing

The scientific testing of the code was carried out over the period August 2005 - May 2006. The main scientific tests involved:

- Monitoring the global innovation statistics obtained using the pre-processed radiances.
- Assessing the impact of assimilating the lower atmospheric temperature sounding radiances, where the accuracy requirements are most stringent.

This testing and the results obtained are described in detail in Section 7 of the *Scientific Description*. The scientific tests are summarised in Table 5 below.

Test	Description	Criteria	Results	Pass/Fail
Scientific test 1: Innovation statistics	Monitor first guess departures (O-B) for global datasets over a period of 2 months or more	Std for LAS channels 2-7 < 0.3K (unprocessed radiances have stdevs of 0.5- 1.0K)	Monitoring over 01/02/06 – 01/05/06 gave STD's of : Ch2 0.235K Ch3 0.179K Ch4 0.156K Ch5 0.272K Ch6 0.474K Ch7 0.465K (see Scientific Description for	Pass for Ch2-5. Fail for channels 6&7. (corrections for Channels 6 & 7 particularly difficult - further development

			more details	required.)
Scientific test 2: Impact on forecast quality	Introduction of SSMIS radiances into Global 3D/4D Var assimilation results in measurable improvement in forecast quality	SH PMSL scores improved over forecast range Day 2-4. RMSE errors reduced	3D-Var global experiment. Winter 2005/2006. Δ (%RMSE) ~ -(0-2.5%)	Pass

Table 5: Summary of scientific tests and results

In summary, the performance of the pre-processor for the mid tropospheric sounding channels 2,3 and 4 is such that computed global innovations (first guess departures) have standard deviations of 0.24K or less, which is comparable with the innovations obtained from AMSU-A data using the Met Office global model. For higher peaking channels the performance is not as good, for reasons that are not fully understood but are probably due to the need for further tuning of the reflector emission corrections. Nevertheless it was decided that the good performance for the tropospheric channels should be sufficient to give positive impact on forecast quality and the second set of scientific tests (forecast impact studies) were carried out. In 3D-Var assimilation trials the assimilation of SSMIS pre-processed radiances gave reductions in RMS forecast errors for Days 2-4 for mean sea level pressure in the Southern Hemisphere of 1 - 2.5% when added on top of assimilation systems including 2 and 3 AMSU-A instruments.

Technical testing

The technical testing of the code was carried out in three parts.

- **Compiler tests** – to ensure cross compiler compatibility
- **Platform tests** – to ensure compatibility across platforms
- **Timing tests** – to ensure the top level product specification is met (*ie* processing time < 10 minutes)

These tests, and the results obtained, are described below:

Compiler Tests

The testing strategy to verify the code would compile and run successfully for different compilers involved generating executables using the Intel 7.1 and the Intel 9.0 compilers. These executables were then used to generate averaged BUFR data from the same set of input BUFR files. As the binary output BUFR files are difficult to compare directly, a utility program was written to decode the output BUFR files and write the brightness temperature data to an external ASCII file. The ASCII files generated from BUFR files generated from different executables can then be compared easily. The process is shown schematically in Figure 1 below:

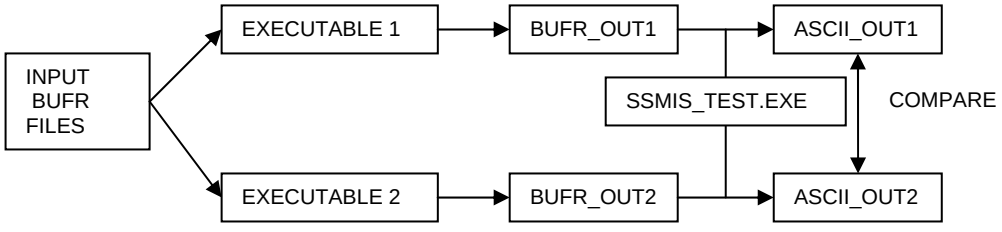


Figure 1: Schematic showing the testing strategy for the inter-compiler tests.

The test output (ASCII files) can be found at :

```
$USER_DIR/SSMIS_PP/data/test_data/ TEST_OUT_IFC.DAT
$USER_DIR/SSMIS_PP/data/test_data/ TEST_OUT_IFORT.DAT
```

The compiler tests are summarised below in Table 6.

Test	Description	Result	Pass/Fail
Compiler test 1	Intel 9.0 (IFORT) vs Intel 7.1 (IFC) test. Code builds successfully	Code built with both compilers	Pass
Compiler test 2	Output from run of builds gives brightness temperature differences < 0.02K	Differences of 0.01K for 1 channel in 1.6% of observations	Pass

Table 6: Summary of Compiler test and results

A `diff` of the resulting ASCII output files showed there to be differences of 0.01K in one of the 24 channels for 1.6% of the observations. This was deemed to be an acceptable level of discrepancy.

Building the test code is described in the section on *Installing and Running the Pre-processor*

Platform testing

The platform tests involved compiling the code on two different platforms:

- A Dell Precision 350 Workstation running Red Hat linux 9.0
- A Dell PowerEdge 2800 serer running Red Hat Enterprise 4 linux

A locally built (using the Intel9.0 compiler) executable on the second platform was used to generate BUFR output files. These were then decoded, using the utility program described above, to generate ASCII files which can then be compared with the ASCII files generated on the first platform.

The corresponding files can be found at:

```
$USER_DIR/SSMIS_PP/data/test_data/TEST_OUT_PLATFORM2.DAT !Platform_2
$USER_DIR/SSMIS_PP/data/test_data/TEST_OUT_IFORT.DAT     !Platform_1
```

The results of the platform tests are described in Table 7 below

Test	Description	Results	Pass/Fail
Platform test 1	Check output from builds on 2 platforms is equivalent . Brightness temperature differences < 0.02K	Output differences zero	Pass

Table 7: Summary of platform tests and results obtained

There were found to be no differences between the results.

Timing tests

SSMIS datasets are normally received one orbit at a time. The mean time between orbits is approximately 110 minutes. For operational NWP systems it is important that this data is made available as quickly as possible to the assimilation system in order the coverage provided by the data is optimal. As specified in the *Product Specification,* the run time for the pre-processor should not exceed 10 minutes. The timing tests and results are described in Table 8 below:

Test	Description	Results	Pass/Fail
Timing test	Run time for pre-processor should be less than 10 minutes	Platform 1 timings: 4- 5 minutes (due to variability in orbit data length) Platform 2 timing: 50 sec -1 minute 10 second	Pass

Table 8: Summary of timing tests and results obtained

Installing and using the SSMIS_Preprocessor

This section describes the steps necessary to install and run the pre-processor.

The steps involved in installing the package are :

- Unpacking the package
- Building the library of BUFR utilities
- Building the SSMIS_Preprocessor executable
- Setting up links to the coefficient files
- Running the tests (optional)

In order to run the package, processing a data stream the following additional steps are required:

- Setting up the scripts which prepare the BUFR files for input to the pre-processor
- Running the pre-processor from cron

Each of these steps are described in more detail below

Unpacking the tar file

The SSMIS pre-processor is available as a tar file :`SSMIS_PP.tar` which is 541 Mbytes in size. The tar file was created using the command:

```
tar cvf SSMIS_PP.tar SSMIS_PP
```

the pre-processor is therefore unpacked using the command

```
tar –xvf SSMIS_PP.tar
```

The directory structure

The directory structure in the tarred package is shown in Figure 2 below:

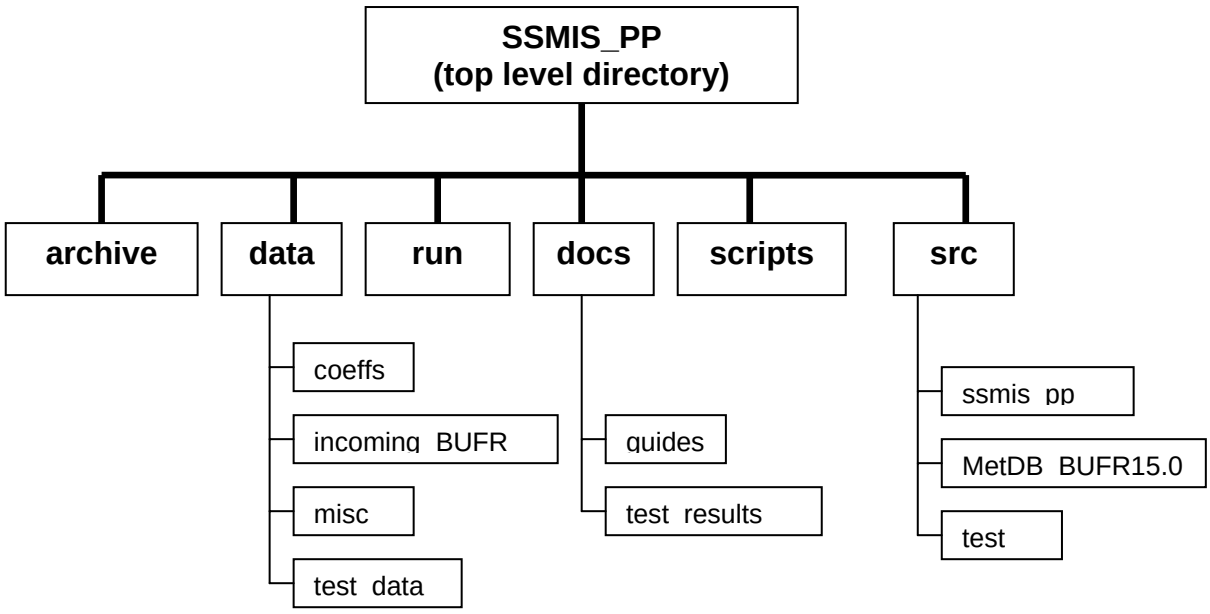


Figure 2: Directory structure of the SSMIS pre-processor.

A brief description of the content of these directories is given below:

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

`/SSMIS_PP/data/coeffs` contains all coefficient data, including averaging coefficients and reflector correction coefficients

`/SSMIS_PP/data/incoming_BUFR` is currently empty but is intended for use as a repository for incoming BUFR files

`/SSMIS_PP/data/test_data` contains the output of the tests described in the section on *Code Testing* as well as the input BUFR data required for the

`/SSMIS_PP/docs/guides` contains both the *Scientific and Technical Descriptions*

`/SSMIS_PP/src/ssmis_pp` contains the source and interface files for the SSMIS pre-processor

`/SSMIS_PP/src/MetDB_BUFR15.0/` contains the source for the BUFR object library
`/SSMIS_PP/src/test` contains the source code for the test utilities

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. An example makefile is provided to do this in the directory, see:

`/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`

Building the SSMIS Preprocessor

A script for building the pre-processor is given at:

`SSMIS_PP/scripts/SSMIS_PP_Build_EXAMPLE.sh`

The paths specified by the variables USERDIR (top level user directory) and EXEDIR (directory for built executable) at the beginning of the build script should be altered according to the users directory structure.

The fortran compiler specified by the variable FC and the fortran compiler flags specified by the variable `Fortran_Flags` may have to be changed according to the compiler selected by the user.

Running this script should produce an executable:

`SSMIS_PP/bin/test_dir/SSMIS_PREPROCESSOR.exe`

There are further comments in the script.

Linking files to the run directory

As a checklist, the following files need to be linked to the run directory in order of the pre-processor to run:

superob_coefficients.dat -> $USER_DIR/SSMIS_PP/data/coeffs/superob_coeffs_sigma50.dat

WORKLIST -> $USER_DIR /SSMIS_PP/data/test_data/WORKLIST

T_ARM_TEMPLATE.DAT -> $USER_DIR /SSMIS_PP/data/coeffs/T_ARM_template_310306.DAT

TB_CORRECTIONCOEFFS.DAT ->
$USER_DIR/SSMIS_PP/data/coeffs/TB_CORRECTIONCOEFFS.DAT

T_CORRECTION_FACTOR.DAT -> $USER_DIR
/SSMIS_PP/data/coeffs/T_CORRECTION_FACTOR.DAT

SSMIS_intrusion_map.dat ->$USER_DIR /SSMIS_PP/data/coeffs/map_data_oct05.dat

TABLEB ->$USER_DIR /SSMIS_PP/data/coeffs/TABLEB

TABLED ->$USER_DIR /SSMIS_PP/data/coeffs/TABLED

env_interpolation_coeffs.dat ->$USER_DIR /SSMIS_PP/data/coeffs/env_interpolation_coeffs.dat

uas_interpolation_coeffs.dat -> $USER_DIR/SSMIS_PP/data/coeffs/uas_interpolation_coeffs.dat

ima_interpolation_coeffs.dat -> $USER_DIR/SSMIS_PP/data/coeffs/ima_interpolation_coeffs.dat

SSMIS_PREPROCESSOR.exe ->$USER_DIR /SSMIS_PP/bin/SSMIS_PREPROCESSOR.exe

In addition, to run the initial acceptance tests, the following example data files should be linked to the run directory:

NPR_TDLB.SA_D06090_S0529_E0714_B1263637_NS ->
$USER_DATA/SSMIS_PP/data/test_data/NPR_TDLB.SA_D06090_S0529_E0714_B1263637_NS

NPR_TDEB.SA_D06090_S0529_E0714_B1263637_NS ->
$USER_DATA/SSMIS_PP/data/test_data/NPR_TDEB.SA_D06090_S0529_E0714_B1263637_NS

NPR_TDIB.SA_D06090_S0529_E0714_B1263637_NS ->
$USER_DATA/SSMIS_PP/data/test_data/NPR_TDIB.SA_D06090_S0529_E0714_B1263637_NS

NPR_TDUB.SA_D06090_S0529_E0714_B1263637_NS ->
$USER_DATA/SSMIS_PP/data/test_data/NPR_TDUB.SA_D06090_S0529_E0714_B1263637_NS

Acceptance tests

Once the code is built and the links set up the acceptance tests can be run to generate an averaged and remapped bufr output file, which can then be decoded and compared against the test output in the test directory.

First of all the Test code should be built. The build script can be found in the scripts directory :

`/SSMIS_PP/scripts/SSMIS_PP_TEST_Build.sh`

The script will need to be edited to change the variables `BUFRDIR`, `SSMISDIR` and `EXEDIR`. On successful running of the script the executable is generated in the `bin` directory. This can then be linked to the `run` directory and the test program (`SSMIS_TEST.exe`) run. The program takes as input a filename read from the file `INPUT_FILE.TXT` (a copy of this file exists in the run directory and currently specifies the filename `LAS_BUFR_FILE_AV.BUFR` which is the default filename produced by running the SSMIS pre-processor.)

`SSMIS_TEST.exe` generates, as output, the ASCII file `TEST_OUT.DAT` This can then be compared with the template test file at:

`$USER_DIR/SSMIS_PP/data/test_data/TEST_OUT_TEMPLATE.DAT`

Running the pre-processor routinely

In order to run the pre-processor routinely it is necessary to prepare the file `WORKLIST` with the names of the four incoming BUFR files as they are 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 a complete set of the four BUFR files for each orbit. If all four files are present, the filenames are copied to the file `WORKLIST`, copied to a run directory and the SSMIS pre-processor is run. A copy of this script is included as *Appendix C* and may be used as a starting point for developing user specific scripts.

Additional scripts will be required to copy the pre-processed files to an archive if desired.

Known bugs & future improvements

- The existing pre-processor does not deal with missing scan lines very well. The remapping and averaging routines do not check for sequential scan line indexing in the working array. Monitoring has been put in place and it appears that missing scan lines are infrequent. When scan lines aare missing from the input data then it is normally only one or two consecutive scan lines. This should not present a serious problem,

- At the moment the intrusion flagged scan lines are flagged using the SSMIS rain flag. It is not possible therefore to differentiate between good quality rain affected observations and those affected by solar intrusions into the warm load in the output BUFR data. This will be rectified by using the additional bits available in the rain flag.

- The evolution of the reflector arm temperature round the orbit is not constant throughout the year as the relative sun angle changes and different parts of the spacecraft obscure the reflector. There is a need, therefore, to periodically update the template file which is required for the generation of the lagged correction to the measured arm temperature. This could be rectified by generating a template file which captures the evolution of the arm temperature throughout the year.

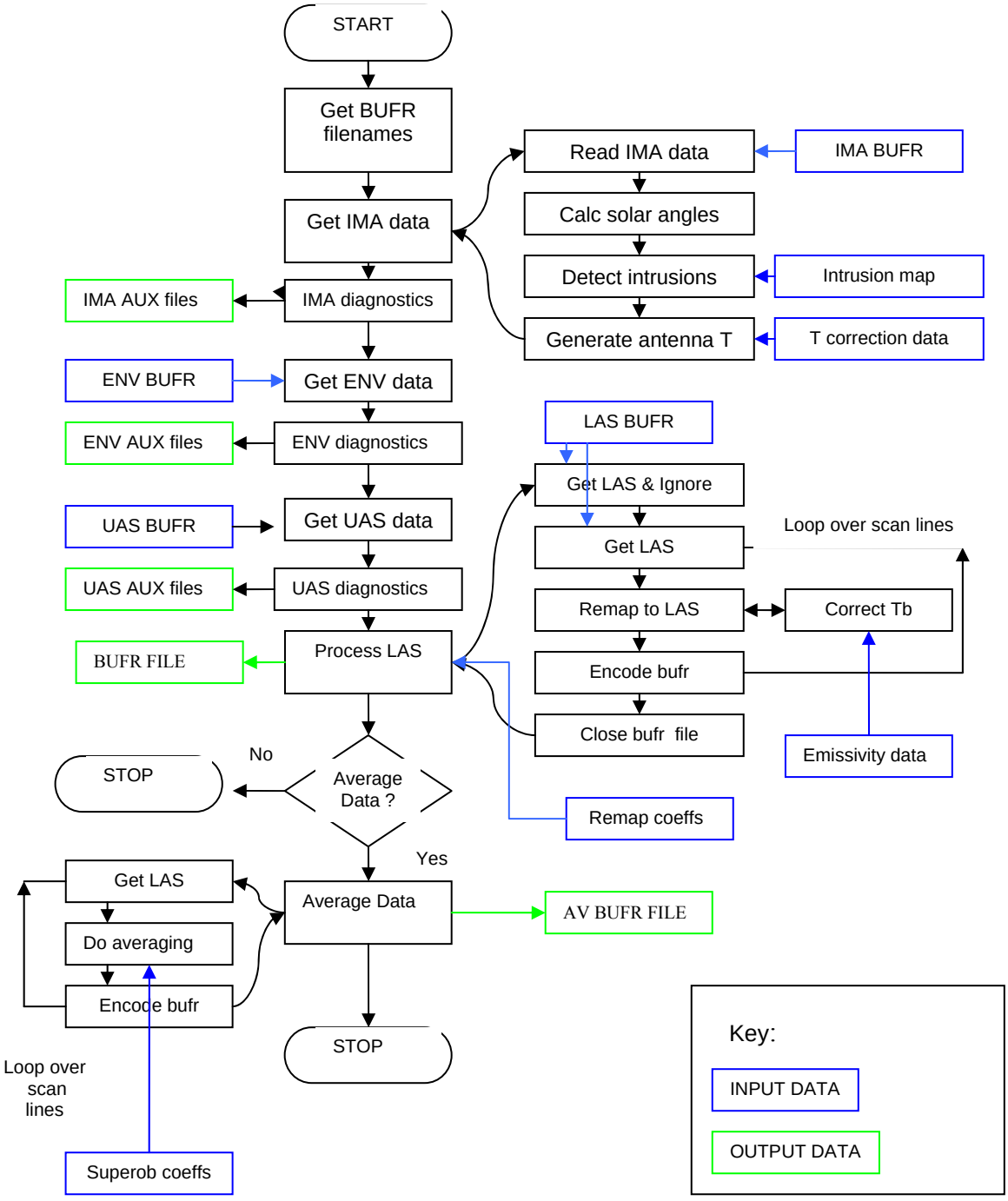
- The solar intrusion map also has to be update periodically. This could be rectified by generating a pre-computed array of intrusion maps which could be read in by the pre-processor and the most appropriate map selected depending on the date.

- There is further work to be done on the reflector emission correction which at present seems to work well for channels 2-4, but less well for higher peaking temperature sounding channels. This is described in more detail in the Scientific Description.

- More comprehensive error trapping will be implemented throughout the code in future versions

- The code is untested to date on Unix platforms. Work on this is underway and a Unix compatible version of the pre-processor will be released in future.

Appendix A. Flowchart of the SSMIS Pre-processor



Appendix B. Calling Tree for the SSMIS preprocessor

```
SSMIS_PREPROCESSOR                          ! main program
        SSMIS_PP_GetBufrfiles
        SSMIS_PP_GenRemapLASBUFR
                SSMIS_PP_GetIMA
                        BUFRREAD_IMA
                        DEBUFR
                        DESFXY
                SSMIS_PP_CalcSolZenAndAz
                        Date32
                SSMIS_PP_DetectSolarIntrusions
                SSMIS_PP_GenTant
                SSMIS_PP_OutputDiagnostics_IMA
                SSMIS_PP_GetENV
                        BUFRREAD_ENV
                        DEBUFR
                        DESFXY
                SSMIS_PP_OutputDiagnostics_ENV
                SSMIS_PP_GetUAS
                        BUFRREAD_UAS
                        DEBUFR
                        DESFXY
                SSMIS_PP_OutputDiagnostics_UAS
                SSMIS_PP_ProcessLAS
                        SSMIS_PP_GetLASAndIgnore
                                BUFRREAD_LAS
                        MetDB_COPEN
                        DATIM
                        SSMIS_PP_GetRemapCoeffs
                        SSMIS_PP_GetTBCorrectionCoeffs
                        SSMIS_PP_GetLAS             ! begin loop over scan lines
                                BUFRREAD_LAS
                                DEBUFR
                        SSMIS_RemaptoLAS
                                DESFXY
                                SSMIS_PP_CorrectTB
                        SSMIS_PP_Encode
                                ENBUFV2
                                METDB_CWRITE
                        MetDB_CCLOSE                ! end loop over scan lines
                        SSMIS_PP_Average
                                METDB_COPEN
                                SSMIS_PP_GetLas_And_Ignore
                                BUFRREAD_LAS
                                SSMIS_GetLAS        ! begin loop over scan lines
                                SSMIS_DoAveraging   ( to generate averaged data)
                                SSMIS_PP_Encode
                                        ENBUFV2
                                        METDB_CWRITE
                                        METDB_CCLOSE ! end loop over scan lines
```

Appendix C. Generating input to SSMIS pre-processor

```
#! /usr/bin/ksh

#-------------------------------------------------------------------------------
#
# Script to generate input for the SSMIS PREPROCESSOR
#
# Purpose:
#
# Examine incoming data directory for new SSMIS BUFR files.
# These files will take the form of 1 file per orbit per data
# type ie, there will be one file for each of UAS, LAS, ENV
# and IMG scans.
#
# Notes:
#
# 1. The user needs to specify the top level directory, via the
# environment variable USER_DIR.
#
# 2. The script checks for the presence of 4 files for a given orbit number,
# and checks that each of the four instrument subtypes is present, before
#  proceeding to process the data
#
# Version    Date    Author   Comments
# ========    ====    =======   =========
#    1.0  21/7/06   W. Bell   New version with clearer commenting and
#                             paths specified relative to user defined
#                             top level directory
#
#-------------------------------------------------------------------------------

#---------------------------------------
# incoming  data directory is
# $USER_DIR/SSMIS_PP/data/incoming_BUFR
# The user should specify the top level
# Directory using the environment variable
# USER_DIR below.
#---------------------------------------

USER_DIR=/data/radsat/frpd        # USER should specify this as required

lodged_string="lodged in metb"

cd ${USER_DIR}/SSMIS_PP/data

#-----------------------------------------------------
# remove previous versions of DIRLIST and  WORKLIST
# from ${USER_DIR}/SSMIS_PP/data
#-----------------------------------------------------


if test -f DIRLIST
then
  rm  DIRLIST
fi


if  test -f WORKLIST
then
echo 'removing WORKLIST file'
rm WORKLIST
fi

#-----------------------------------------------------
# remove previous versions of DIRLIST and  WORKLIST
```

```
# from ${USER_DIR}/SSMIS_PP/data/incoming_BUFR
#-------------------------------------------------

if test -f ${USER_DIR}/SSMIS_PP/data/incoming_BUFR/DIRLIST
then
 echo 'removing old DIRLIST file from incoming_BUFR directory'
 rm ${USER_DIR}/SSMIS_PP/data/incoming_BUFR/DIRLIST
fi

#-------------------------------------------------------------------------------
# Send a 1 column list of the incoming data directory to a file
# DIRLIST.  Files are listed most recent first so if 3 files for a given orbit are
# present, but not the fourth, then the script will do nothing until another orbit
# arrives and the four files are checked for.
#-------------------------------------------------------------------------------

cd ${USER_DIR}/SSMIS_PP/data/incoming_BUFR

ls -1t *NS > DIRLIST
cd ${USER_DIR}/SSMIS_PP/data
mv ${USER_DIR}/SSMIS_PP/data/incoming_BUFR/DIRLIST .

#-------------------------------------------------------------------------
# strip the orbit number, and extension  (eg R45426.BUFR) from the
# first file in this DIRLIST, and store to the variable 'orbit_number'
#-------------------------------------------------------------------------

field1=$(cut -d_ -f4 DIRLIST | head -1)
field2=$(cut -d_ -f5 DIRLIST | head -1)
orbit_number=$(cut -d_ -f6 DIRLIST | head -1)

if [ $? -eq 0 ]
then
echo 'Orbit number and extension is:'
echo $orbit_number
echo $field1
echo $field2
fi

#-----------------------------------------------------------------
# Now check how many of these files there are, store in numfiles
#-----------------------------------------------------------------

numfiles=$(ls ${USER_DIR}/SSMIS_PP/data/incoming_BUFR/*$field1*$field2*$orbit_number* |wc -l)

echo 'Number of files for this orbit number is :'
echo $numfiles

#-------------------------------------------------------------------------
# now check there are exactly 4, if so then copy to a working directory
# and run executable which de-codes them, interpolates and  re-encodes
# them
#-------------------------------------------------------------------------

if [ $numfiles -eq 4 ]
 then


mv                              ${USER_DIR}/SSMIS_PP/data/incoming_BUFR/*$field1*$field2*$orbit_number*
${USER_DIR}/SSMIS_PP/run

#-------------------------------------------------------------------------
# sometimes there are more than 4 files per orbit, typically $field1 and
# $field2 are the same for sets of 4, therefore the most recently received
# complete batch of 4 is selected for subsequent processing by the line
# above.  The other files are copied directly to the archive directory by
# the line below.
#-------------------------------------------------------------------------
```

```
echo ' Four latest files moved to run directory '
echo ' Additional files present in directory : ?'

ls -1 ${USER_DIR}/SSMIS_PP/data/incoming_BUFR/*$orbit_number*

if [$? -eq 0]
then

mv ${USER_DIR}/SSMIS_PP/data/incoming_BUFR/*$orbit_number*  ${USER_DIR}/SSMIS_PP/archive

fi


cd  ${USER_DIR}/SSMIS_PP/run

#----------------------------
# remove last WORKLIST file
#----------------------------

rm WORKLIST

#-----------------------------------------
# Now check for the presence of 1 of each
# of the LAS,UAS,ENV and IMA files by
# checking a listing of each filetype returns
# a zero, then add up the exit codes
# and check it sums to zero before
# proceeding
#-----------------------------------------

ls -1 NPR_TDUB*$orbit_number*
a=$?
ls -1 NPR_TDLB*$orbit_number*
b=$?
ls -1 NPR_TDIB*$orbit_number*
c=$?
ls -1 NPR_TDEB*$orbit_number*
d=$?


e=`expr $a "+" $b "+" $c "+" $d`


if test $e = 0
then

#-------------------------------------------------
# if all are present then copy these 4 filenames
# to a file WORKLIST
#-------------------------------------------------


ls -1 NPR_TDUB*$orbit_number* > WORKLIST
ls -1 NPR_TDLB*$orbit_number* >> WORKLIST
ls -1 NPR_TDIB*$orbit_number* >> WORKLIST
ls -1 NPR_TDEB*$orbit_number* >> WORKLIST

#-----------------------------------------------------------
# Now check for the existence of any BUFR files in this
# directory. If present remove
#-----------------------------------------------------------

ls -lt *.BUFR

if [$? -eq 0]
then
rm *.BUFR
fi
```

```
#-------------------------------------------------
# Now run executable (input taken from WORKLIST)
#-------------------------------------------------

#----------------
# log start time
# in logfile
#----------------

date >> LOGFILE

./SSMIS_PREPROCESSOR.exe


#-------------------------------------------------
# Preprocessor generates diagnostic output which
# is concatenated with existing output
#-------------------------------------------------

cat IMA_DIAGNOSTICS_AUX.DAT >> DIAGNOSTICS_AUX.DAT
cat IMA_DIAGNOSTICS_SL.DAT >> DIAGNOSTICS_SL.DAT


            if test -f  LAS_BUFR_FILE_AV.BUFR

            then

          #add to log file date and name of LAS file
          date >> LOGFILE
          ls -lt NPR_TDLB*$orbit_number* >> LOGFILE


               mv LAS_BUFR_FILE_AV.BUFR LAS_BUFR_AV_$orbit_number.BUFR
          ls -lt LAS_BUFR_AV_$orbit_number.BUFR >> LOGFILE



#-----------------------------------------------------------------

mv NPR_TDUB*$orbit_number* ${USER_DIR}/SSMIS_PP/archive
mv NPR_TDIB*$orbit_number* ${USER_DIR}/SSMIS_PP/archive
mv NPR_TDLB*$orbit_number* ${USER_DIR}/SSMIS_PP/archive
mv NPR_TDEB*$orbit_number* ${USER_DIR}/SSMIS_PP/archive
mv LAS_BUFR_AV_$orbit_number.BUFR ${USER_DIR}/SSMIS_PP/archive


fi



  else

echo " not all 4 files present "

  fi
else

echo " $numfiles does not equal 4 !"
echo " therefore processing stopped "

fi
```

Appendix D. Version Control

The SSMIS pre-processor and all associated code has been place under version control using *Subversion*, the Met Office standard revision control system which permits parallel code development using a 'Copy-Modify-Merge' model for versioning.

Excellent documentation on *Subversion* can be found at:

http://subversion.tigris.org/

(see the link to the *Subversion Book* from this page)

The SSMIS pre-processor and all associated documentation and coefficient files, test files etc, are held in an svn repository at:

`file:///home/fr1200/frwb/svn/SSMIS_PP_repository`

(this is a local disk on the Met Office linux network and is not accessible to external users.)

For example to create a working copy of the code in a directory `$USER_DIR/SSMIS_PP` :

```
cd $USER_DIR
svn checkout file:///home/fr1200/frwb/svn/SSMIS_PP_repository SSMIS_PP
+
```