ALADIN Project Stay report – version 2.0 (September 2016)

Author: Maria Monteiro (Instituto Português do Mar e da Atmosfera, IPMA I.P.)
In collaboration with Alena Trojakova , Czech Hydrometeorological Institute (CHMI)
Period: 11-15 January 2016

Topic title: Validation of a back-phased version of the source code BATOR

1 Introduction

This document reports the activities done during a short stay at CHMI on the validation of a back-phased version of BATOR. Generally speaking, this tool is used inside the ALADIN community to read observational data and convert it into a data base, the Observational DataBase (ODB), which manages the observational data along an assimilation cycle. The BATOR source code used during this stay has been provided by Frank Guillaume (from Météo-France) in September 2015 after being back-phased from CY41T1 to CY38T1 [1]. The main goal was to test and validate the tool under a local operational environment (outside Météo-France framework) to insure it could be a solution to digest WMO Binary Universal Form for the Representation (BUFR) of conventional data which is actually disseminated via the Global Telecommunication System (GTS). SYNOP and TEMP observations have been used during the tests.

WMO is promoting an upgrade on the GTS dissemination procedures which involves a change of the transmission code forms. Namely, the use of Table Driven Code Forms (TDCF) like BUFR, will be adopted for the transmission of observational data instead of the Traditional Alphanumeric Codes (TAC). Due to this fact, some countries may need (or see it as an opportunity) to change the observational data initialization procedures on their local NWP systems. For instance, GTS Spanish observational data is now strictly available under BUFR and Portugal is not able to digest it yet.

Inside the ALADIN consortium, Météo-France (MF) is already able to directly use the WMO BUFR format for conventional data (through BATOR) since its internal cycle CY38, but most of the ALADIN members' local data assimilation systems still rely on a conversion from the ASCII OBSOUL data format [see Table https://docs.google.com/spreadsheets/d/1_lQMFDaRRDNEng21asHKQ42Hx_-llOnRG3EX3BtfzVo/edit#gid=1092978226 from ACNA (Maria Derkova), as mentioned by 2]. The idea is then to take advantage of the developments already done by MF. However, at the moment of the writing of this report MF source code is already under the cycle version CY41T1 while most of the remaining ALADIN countries run operationally a previous cycle (MF BATOR compatible with WMO BUFR templates will only be exported from CY41 onwards).

To take advantage of recent developments in BATOR one should back-phase it from the latest available version to the one which is validated locally. There are two main reasons to adopt this methodology : a) the WMO templates are not fully stabilized (they have been evolved since) and may look differently from country to country (under certain rules); and b) the conversion to an unnecessary ASCII format (OBSOUL) may bring an extra source of error and need of further maintenance (MF will not support/maintain OULAN tool in the

future).

This stay has encompassed the following steps:

1. merging back-phased BATOR with its local version (PART I: SYNOP and PART II: TEMP);
2. compiling and testing the source code with recent data (PART I: SYNOP and PART II: TEMP);
3. repeating the same exercise 2) on the Portuguese operational environment.

In the following sections the main achievements on each of the different topics are reported. Note that in this second version of the report new information has been included which concerns the work related to TEMP messages (PART II on each section); besides, the information on SYNOP messages has been completed.

2 Merging BATOR with CHMI (CZ) local version.

For this exercise, we have used the locally (CZ) validated version of BATOR CY38T1tr_op4 as the reference. The methodology followed the steps:

i) identification of the changes between the local and MF's BATOR CY41toCY38 version
ii) back-phasing of BATOR changes to the local version of this source (including its compilation and creation of the new executable).

These steps have been repeated in sequence for the two different types of meteorological observation: first to the SYNOP and second TEMP.

For information, the back-phased MF BATOR CY41toCY38 version for these two types of observations included the re-visit of the following set of sources (located in odb/pandor/module):

bator_datetime_mod.F90
bator_decodbufr_mod.F90
bator_ecritures_mod.F90
bator_init_mod.F90
bator_lectures_mod.F90
bator_module.F90
bator_saisies_mod.F90

The corresponding source code can be found at Météo-France computing platforms in beaufix, at the path:

/home/gmap/mrpe/monteiro/SAVE/src/batorodb/phasing/src/local/odb/pandor/module

or

/home/gmap/mrpe/monteiro/public/bator.

PART I: SYNOP

Since, for SYNOP, the truly different piece of code - which deals with the different types of observations - is bator_decodbufr_mod.F90, the comparison between the local (CZ) and MF's versions has driven us to the following set of merged modules:

bator_module.F90
bator_decodbufr_mod.F90
bator_datetime_mod.F90

The tables A.1 to A.3 in Appendix A register in detail the main code changes. In blue are represented the added features to the local version.

We noted that the main changes were due to:

a) introduction of new functions for dates manipulation;
b) introduction of a few features related to the French SYNOP stations;
c) introduction of new controls related to the observation country;
d) corrections to the total cloudiness algorithm; and, as a consequence
e) the introduction of new global variables.

PART II: TEMP

The comparison between the local (CZ/PT) and MF's versions of code has driven us to the following set of merged modules (in addition to the changes done for SYNOP):

bator_module.F90
bator_decodbufr_mod.F90

The tables B.1 and B.2 in Appendix B register in detail the main code changes. In blue are represented the added features to the local version.

In addition to the changes found for SYNOP, now the main differences were due to:

a) recognition of different radiosonde WMO BUFR templates (regional changes); and as consequence
b) the introduction of new global variables.

3 Compiling, testing and validating the source code with recent data.

PART I: SYNOP

In order to test the new executable of BATOR different data sets have been used. The following Table describes them:

Table 1 – Data sets used to test the back-phased BATOR

FILENAME	GTS dissemination center	GTS reception center	Number of BUFR messages	Number of observations (total number of subsets)	Data date.time(UTC)
00000025	AEMET (SP)	IPMA (PT)	1	1	20160107.12
00000026	IPMA (PT)	IPMA (PT)	1	1	20160107.12
ISMD01_OKPR_100000_547	CHMI (CZ)	CHMI (CZ)	1	7	20160110.00
ISMA45_LPMG_100000_704	IPMA (PT)	IPMA (PT)	1	29	20160110.00
bufrpt_2016011300	IPMA (PT) AEMET (SP)	IPMA (PT)	30	180	20160113.12 20160113.13

For this exercise it was useful to have a diverse set of samples of GTS BUFR messages, since by comparative analysis we were lead many times to useful conclusions. In particular, we had files with just one and with several BUFR messages, that in turn concerned just one individual SYNOP observation (uncompressed) or several SYNOP observations (compressed).

In order to validate the new executable, the content of the observational messages was checked in different moments of the BATOR processing chain. In particular, messages contents were verified after decoding and before delivering them to BATOR and after BATOR, already under the ODB format. To apply this methodology the following tools have been used (see also Appendix C).

As general conclusions it was found that:

a) the back-phased BATOR was able to read the BUFR messages and to process them properly;

b) BATOR can abort if GTS data is given as input exactly as it is received due to the presence of wrongly formated data. For instance in the file bufrpt_2016011300 it was possible to find at least 3 observations (from stations: 08094, 08160 and 60015) where lat/lon information was missing, most probably because a Retard Message will be later disseminated but it was not ready at the moment[1] of the first dissemination. This means that data should always come from a pre-processing which could act as an intermediate metadata quality control step, like a dedicate pre-processing or a data-base.

c) the param.cfg used was the one which comes with the export version to ensure that an input BUFR contains only single data for each SYNOP station with meaningful values.

1 According to Portuguese GTS expert.

PART II: TEMP

The code source for the TEMP messages is still under validation.

4 Testing the source code in real-time in Portugal.

To test the back-phased source code in Portugal it was necessary to build at the first place the working environment. Appendix D registers the main steps.

5 Acknowledgments

Acknowledgments are due to Frank Guillaume for providing the BATOR CY41toCY38 version of the code; to the CHMI colleagues for the friendship environment and concerns during my stay in Prague.

Bibliography

[1] Monteiro, M., Surface representation on the Portuguese model and OI_MAIN cycling, ALADIN Project Stay report, 21-25 September 2015, GMAP/CNRM, France

[2] ALADIN LTM Meeting Minutes, Wednesday 15th April 2015, 15:45-18:00 Elsinore, Denmark, http://www.cnrm-game-meteo.fr/aladin/IMG/pdf/minutes_ltm19.pdf

Appendix A: SYNOP back-phasing

Table A.1 – Main changes performed on the source code **bator_module.F90** for SYNOP

Local (CZ) version	MF's version	Back-phased version
None	! for solomm data LOGICAL :: LPacome INTEGER(KIND=jpim),DIMENSION(20) :: Origine	! for solomm data LOGICAL :: LPacome INTEGER(KIND=jpim),DIMENSION(20) :: Origine

Table A.2 – Main changes performed on the source code **bator_datetime_mod.F90** for SYNOP

Local (CZ) version	MF's version	Back-phased version
USE PARKIND1, ONLY : jprb	USE PARKIND1, ONLY : jprb,jpib	USE PARKIND1, ONLY : jprb,jpib
PUBLIC :: New_DateFromArray, Return_DateArray, Add_Date	PUBLIC :: New_DateFromArray, Return_DateArray, Add_Date, Add_BigDate, Diff_Date	PUBLIC :: New_DateFromArray, Return_DateArray, Add_Date, Add_BigDate, Diff_Date
None	FUNCTION Add_BigDate(date,seconds)	FUNCTION Add_BigDate(date,seconds)
None	FUNCTION Diff_Date(date1, date2)	FUNCTION Diff_Date(date1, date2)

Table A.3 – Main changes performed on the source code **bator_decobufr_mod.F90** for SYNOP

Local (CZ) version	MF's version	Back-phased version
SUBROUTINE Synop(kobs,kw,kel,tconfig,iterr,j,jump1,jump2,jump3,jump4,jump5,iter_rr,l_ship)	SUBROUTINE Synop(kobs,kw,kel,tconfig,iterr,j,jump1,jump2,jump3,jump4,jump5,iter_rr,l_ship, l_radome)(*)	SUBROUTINE Synop(kobs,kw,kel,tconfig,iterr,j,jump1,jump2,jump3,jump4,jump5,iter_rr,l_ship, l_radome)(*)
	CALL Synop(kobs,kw,kel,tconfig,iterr,j,jump(...,...),0,jump(...,...),0,jump(...,...),iter_rr,l_ship,l_radome)	CALL Synop(kobs,kw,kel,tconfig,iterr,j,jump(2,2),0,jump(2,2),0,jump(...,...),iter_rr,l_ship,l_radome)
INTEGER(KIND=jpim) :: i_indic_omm, inddate, i_TypeSta, i_ammend, i_cod	INTEGER(KIND=jpim) :: i_indic_omm, i_pays_orig, inddate, i_TypeSta, i_ammend, i_cod	INTEGER(KIND=jpim) :: i_indic_omm, i_pays_orig, inddate, i_TypeSta, i_ammend, i_cod
None	CHARACTER(len=80) :: chaine LOGICAL :: l_synor INTEGER(kind=jpim), DIMENSION(20) :: PlatformList ! a deplacer en variable 'module' INTEGER(kind=jpim), DIMENSION(5) :: FrenchStates	CHARACTER(len=80) :: chaine LOGICAL :: l_synor INTEGER(kind=jpim), DIMENSION(20) :: PlatformList ! a deplacer en variable 'module' INTEGER(kind=jpim), DIMENSION(5) :: FrenchStates

IF (NINT(values(j*kel+tconfig %iindice(001101))) /= NINT(values(j*kel+tconfig %iindice(001102)))) THEN
i_indic_omm = NINT(values(j*kel+tconfig %iindice(001101)))*10000+NINT(values(j*kel+tconfig%iindice(001102)))
ELSE
i_indic_omm = NINT(values(j*kel+tconfig %iindice(001102)))
ENDIF
	i_pays_orig =NINT(values(j*kel+tconfig %iindice(001101)))
	i_indic_omm=NINT(values(j*kel+tconfig %iindice(001102)))
		i_pays_orig= NINT(values(j*kel+tconfig %iindice(001101)))
		i_indic_omm=NINT(values(j*kel+tconfig %iindice(001102)))

None	!FGFG probleme des radome et synor
	IF (l_radome) THEN
	IF (ksec1(7) == 50) THEN
	l_synor = .TRUE.
	l_radome = .FALSE.
	ELSE
	SELECT CASE (LPacome)
	CASE (.TRUE.)
	IF (ANY(FrenchStates(:) .EQ. i_pays_orig) .AND. .NOT.ANY(Origine(:) .EQ. ksec1(7))) THEN
	i_valid = 1
	iterr(15) = iterr(15) + 1
	ENDIF
	CASE (.FALSE.)
	IF (ANY(FrenchStates(:) .EQ. i_pays_orig)) THEN
	i_valid = 1
	iterr(14) = iterr(14) + 1
	ENDIF
	END SELECT
	ENDIF
		!FGFG probleme des radome et synor
		IF (l_radome) THEN
		IF (ksec1(7) == 50) THEN
		l_synor = .TRUE.
		l_radome = .FALSE.
		ELSE
		SELECT CASE (LPacome)
		CASE (.TRUE.)
		IF (ANY(FrenchStates(:) .EQ. i_pays_orig) .AND. .NOT.ANY(Origine(:) .EQ. ksec1(7))) THEN
		i_valid = 1
		iterr(15) = iterr(15) + 1
		ENDIF
		CASE (.FALSE.)
		IF (ANY(FrenchStates(:) .EQ. i_pays_orig)) THEN
		i_valid = 1
		iterr(14) = iterr(14) + 1
		ENDIF
		END SELECT
		ENDIF

!FGFG problème des plateformes
if (tconfig%iindice(001015) /= -1) then
toto=cvals(nint(values(j*kel+tconfig
%iindice(001015)))/1000)(1:80)
if (index(toto,'Platform') > 0 .OR.
index(toto,'platform') > 0 .OR.
index(toto,'PLATFORM') > 0) then
print *, "platform in name for
",i_indic_omm, s_ident
l_ship = .TRUE.
endif
endif
	!FGFG probleme des plateformes
	if (tconfig%iindice(001015) /= -1) then
	chaine =
	cvals(nint(values(j*kel+tconfig
	%iindice(001015)))/1000)(1:80)
	if (index(chaine,'Platform') > 0 .OR.
	index(chaine,'platform') > 0 .OR.
	index(chaine,'PLATFORM') > 0) then
	l_ship = .TRUE.
	endif
	endif
	if (ANY(PlatformList(:) .EQ. i_indic_omm))
	THEN
	l_ship = .TRUE.
	endif
		!FGFG probleme des plateformes
		if (tconfig%iindice(001015) /= -1) then
		chaine =
		cvals(nint(values(j*kel+tconfig
		%iindice(001015)))/1000)(1:80)
		if (index(chaine,'Platform') > 0 .OR.
		index(chaine,'platform') > 0 .OR.
		index(chaine,'PLATFORM') > 0) then
		l_ship = .TRUE.
		endif
		endif
		if (ANY(PlatformList(:) .EQ. i_indic_omm))
		THEN
		l_ship = .TRUE.
		endif

! --- recuperation du type de station
i_TypeSta = 3
i_cod = 11
IF (values(j*kel+tconfig
%iindice(002001)) /= rabsi) THEN
i_TypeSta =
NINT(values(j*kel+tconfig
%iindice(002001)))
IF (i_TypeSta == 0) i_cod = 14
ENDIF
IF (l_ship) i_cod = i_cod + 10
	! --- recuperation du type de station
	i_TypeSta = 3
	i_cod = 11
	IF (values(j*kel+tconfig
	%iindice(002001)) /= rabsi) THEN
	i_TypeSta =
	NINT(values(j*kel+tconfig%iindice(002001)))
	IF (i_TypeSta == 0) i_cod = 14
	ENDIF
	IF (l_synor) i_cod = 15
	IF (l_radome) i_cod = 16
	IF (l_ship) i_cod = i_cod + 10
		! --- recuperation du type de station
		i_TypeSta = 3
		i_cod = 11
		IF (values(j*kel+tconfig
		%iindice(002001)) /= rabsi) THEN
		i_TypeSta = NINT(values(j*kel+tconfig
		%iindice(002001)))
		IF (i_TypeSta == 0) i_cod = 14
		ENDIF
		IF (l_synor) i_cod = 15
		IF (l_radome) i_cod = 16
		IF (l_ship) i_cod = i_cod + 10

zent(kobs,NCIRLN) = ktdlst(1) ! type de codage bufr	zent(kobs,NCIRLN) = 1 ! bufrtype de type solomm zentsup(kobs,1) = i_pays_orig	zent(kobs,NCIRLN) = 1 ! bufrtype de type solomm zentsup(kobs,1) = i_pays_orig
None	CASE (90000) i_Prescd = 9	CASE (90000) i_Prescd = 9
zwagon(kw,4) = values(j*kel+tconfig%iindice(010009))	zwagon(kw,4) = values(j*kel+tconfig %iindice(010009)) * RG	zwagon(kw,4) = values(j*kel+tconfig %iindice(010009)) * RG

! --- recuperation et controle de
nebulosite totale
Nebul = nabso
IF (values(j*kel+tconfig
%iindice(020010)+jump4) /=rabsi) THEN
Nebul = NINT(values(j*kel+tconfig
%iindice(020010)+jump4))
kw = kw + 1
zwagon(kw,1) = 91
zwagon(kw,2) = Pres
zwagon(kw,3) = rabso
zwagon(kw,4) = Nebul
zwagon(kw,5) = 2048
ENDIF
	! --- recuperation et controle de
	nebulosite totale
	IF (tconfig%iindice(020010) /= -1) THEN
	Nebul = nabso
	IF (values(j*kel+tconfig
	%iindice(020010)+jump4) /=rabsi) THEN
	Nebul = NINT(values(j*kel+tconfig
	%iindice(020010)+jump4))
	kw = kw + 1
	zwagon(kw,1) = 91
	zwagon(kw,2) = Pres
	zwagon(kw,3) = rabso
	zwagon(kw,4) = Nebul
	zwagon(kw,5) = 2048
	ENDIF
		! --- recuperation et controle de
		nebulosite totale
		IF (tconfig%iindice(020010) /= -1) THEN
		Nebul = nabso
		IF (values(j*kel+tconfig
		%iindice(020010)+jump4) /=rabsi) THEN
		Nebul = NINT(values(j*kel+tconfig
		%iindice(020010)+jump4))
		kw = kw + 1
		zwagon(kw,1) = 91
		zwagon(kw,2) = Pres
		zwagon(kw,3) = rabso
		zwagon(kw,4) = Nebul
		zwagon(kw,5) = 2048
		ENDIF

INTEGER(KIND=jpim) :: i, imin	INTEGER(KIND=jpim) :: i, imin, ignored	INTEGER(KIND=jpim) :: i, imin, ignored

CASE DEFAULT
DO i=1,INbTypeObs
imin = MIN(ktdlen, SIZE(tconfig(i)
%ictrlcodage))
IF (SIZE(tconfig(i)%ictrlcodage) ==
ktdlen .AND. ALL(tconfig(i)
%ictrlcodage(1:imin) .EQ. ktdlst(1:imin)))
THEN
NoConfig = i
iret = 0
WRITE(batout,'(">>>>> INFO :
Selected template ",I3," -> ",A16)')
NoConfig,tconfig(NoConfig)%ssensor
EXIT
ENDIF
	CASE DEFAULT
	DO i=1,INbTypeObs
	ignored = COUNT(tconfig(i)
	%ictrlcodage(:) .LT. 0)
	imin = MIN(ktdlen, SIZE(tconfig(i)
	%ictrlcodage) - ignored)
	IF (SIZE(tconfig(i)%ictrlcodage) ==
	ktdlen .AND. ALL(tconfig(i)
	%ictrlcodage(1:imin) .EQ. ktdlst(1:imin)))
	THEN
	NoConfig = i
	iret = 0
	WRITE(batout,'(">>>>> INFO :
	Selected template ",I3," -> ",A16)')
	NoConfig,tconfig(NoConfig)%ssensor
	EXIT
	ENDIF
		CASE DEFAULT
		DO i=1,INbTypeObs
		ignored = COUNT(tconfig(i)
		%ictrlcodage(:) .LT. 0)
		imin = MIN(ktdlen, SIZE(tconfig(i)
		%ictrlcodage) - ignored)
		IF (SIZE(tconfig(i)%ictrlcodage) ==
		ktdlen .AND. ALL(tconfig(i)
		%ictrlcodage(1:imin) .EQ. ktdlst(1:imin)))
		THEN
		NoConfig = i
		iret = 0
		WRITE(batout,'(">>>>> INFO :
		Selected template ",I3," -> ",A16)')
		NoConfig,tconfig(NoConfig)%ssensor
		EXIT
		ENDIF

CASE
('synop_1','synop_2','synop_3','synop_4','
synop_5','synop_6','synop_7')
IF (iterr(10) > 0)
WRITE(batout,'(5X,"WARNING - Pmer nulle :
",8X,I9)') iterr(10)
IF (iterr(11) > 0)
WRITE(batout,'(5X,"WARNING - durées RR
douteuses : ",2X,I9)') iterr(11)
IF (iterr(12) > 0)
WRITE(batout,'(5X,"WARNING - Pb
identifiant OMM : ",4X,I9)') iterr(12)
	CASE
	('synop_1','synop_2','synop_3','synop_4','
	synop_5','synop_6','synop_7')
	IF (iterr(10) > 0)
	WRITE(batout,'(5X,"WARNING - Pmer nulle :
	",8X,I9)') iterr(10)
	IF (iterr(11) > 0)
	WRITE(batout,'(5X,"WARNING - durees RR
	douteuses : ",2X,I9)') iterr(11)
	IF (iterr(12) > 0)
	WRITE(batout,'(5X,"WARNING - Pb
	identifiant OMM : ",4X,I9)') iterr(12)
	IF (iterr(13) > 0)
	WRITE(batout,'(5X,"WARNING - Pression
	nulle : ",4X,I9)') iterr(13)
	IF (iterr(14) > 0)
	WRITE(batout,'(5X,"WARNING - rejet radomeh
	: ",4X,I9)') iterr(14)
	IF (iterr(15) > 0)
	WRITE(batout,'(5X,"WARNING - rejet origine
	: ",4X,I9)') iterr(15)
		CASE
		('synop_1','synop_2','synop_3','synop_4','s
		ynop_5','synop_6','synop_7')
		IF (iterr(10) > 0)
		WRITE(batout,'(5X,"WARNING - Pmer nulle :
		",8X,I9)') iterr(10)
		IF (iterr(11) > 0)
		WRITE(batout,'(5X,"WARNING - durees RR
		douteuses : ",2X,I9)') iterr(11)
		IF (iterr(12) > 0)
		WRITE(batout,'(5X,"WARNING - Pb identifiant
		OMM : ",4X,I9)') iterr(12)
		IF (iterr(13) > 0)
		WRITE(batout,'(5X,"WARNING - Pression nulle
		: ",4X,I9)') iterr(13)
		IF (iterr(14) > 0)
		WRITE(batout,'(5X,"WARNING - rejet radomeh
		: ",4X,I9)') iterr(14)
		IF (iterr(15) > 0)
		WRITE(batout,'(5X,"WARNING - rejet origine
		: ",4X,I9)') iterr(15)

*traitement particulier des radomes francais

Appendix B: TEMP back-phasing

Table B.1 – Main changes performed on the source code **bator_module.F90** for TEMP

Local (CZ) version	MF's version	Back-phased version
! for tempomm data INTEGER(KIND=jpim) :: NbTempMaxLevels LOGICAL :: TempSondOrTraj	! for tempomm data INTEGER(KIND=jpim) :: NbTempMaxLevels, NbMinLevelHr, NFREQVERT_TPHR LOGICAL :: TempSondOrTraj, TempSondSplit, ElimTemp0, ElimPilot	! for tempomm data INTEGER(KIND=jpim) :: NbTempMaxLevels, NbMinLevelHr, NFREQVERT_TPHR LOGICAL :: TempSondOrTraj, TempSondSplit, ElimTemp0, ElimPilot0

Table B.2 – Main changes performed on the source code **bator_decobufr_mod.F90** for TEMP

Local (CZ) version	MF's version	Back-phased version
USE BATOR_UTIL_MOD, ONLY : BATOR_FILTER_RADAR, BATOR_RADAR_WIND_CLEANER, & & VerifDate, UVCOM	USE BATOR_UTIL_MOD, ONLY : BATOR_FILTER_RADAR, BATOR_RADAR_WIND_CLEANER, & & VerifDate, UVCOM, bator_formdate	USE BATOR_UTIL_MOD, ONLY : BATOR_FILTER_RADAR, BATOR_RADAR_WIND_CLEANER, & & VerifDate, UVCOM, bator_formdate
TYPE Header ... REAL(KIND=jprb) :: latitude REAL(KIND=jprb) :: longitude REAL(KIND=jprb) :: Tmer END TYPE Header	TYPE Header ... INTEGER(KIND=jpim) :: FcqInfo ! pour fcqodb	INTEGER(KIND=jpim) :: FcqInfo ! pour fcqodb

CASE ('temp')
IF (iterr(10) > 0) WRITE(batout,'(5X,"WARNING - No levels : ",17X,I9)') iterr(10)
	CASE ('temp')
	IF (iterr(10) > 0) WRITE(batout,'(5X,"WARNING - No levels : ",17X,I9)') iterr(10)
	IF (iterr(11) > 0) WRITE(batout,'(5X,"WARNING - level out : ",17X,I9)') iterr(11)
		CASE ('temp') (PrintMessages)
		IF (iterr(10) > 0) WRITE(batout,'(5X,"WARNING - No levels : ",17X,I9)') iterr(10)
		IF (iterr(11) > 0) WRITE(batout,'(5X,"WARNING - level out : ",17X,I9)') iterr(11)

IdRadiosonde = 15 TypeMessage = 6 SousTypeMessage = (/32,33,34/)	IdRadiosonde = 15 TypeMessage = 6 SousTypeMessage = (/32,33,132/)	IdRadiosonde = 15 (PilotPa PilotM) TypeMessage = 6 SousTypeMessage = (/32,33,132/)
INTEGER(KIND=jpim) :: inddate, iflag_omm, iflag_odb INTEGER(KIND=jpim) :: iobs0,j,i,iretalloc REAL(KIND=jprb) :: zhook_handle	INTEGER(KIND=jpim) :: inddate, iflag_omm, iflag_odb, NbValidLevel,FcqInfo INTEGER(KIND=jpim) :: iobs0,j,i,iretalloc,i1,i2,iz REAL(KIND=jprb) :: zhook_handle,zz1,zz2,zdeltaz,ZLimit REAL(KIND=jprb), DIMENSION(103) :: zatmstd REAL(KIND=jprb), DIMENSION(103) :: zatmstd	INTEGER(KIND=jpim) (Radiosondage) :: inddate, iflag_omm, iflag_odb, NbValidLevel,FcqInfo INTEGER(KIND=jpim) :: iobs0,j,i,iretalloc,i1,i2,iz REAL(KIND=jprb) :: zhook_handle,zz1,zz2,zdeltaz,ZLimit REAL(KIND=jprb), DIMENSION(103) :: zatmstd REAL(KIND=jprb), DIMENSION(103) :: zatmstd
none	INTEGER(KIND=jpim) :: force_profil INTEGER(KIND=jpim), DIMENSION(:),ALLOCATABLE :: ielimine,ilayer INTEGER(KIND=jpim), DIMENSION(12), PARAMETER :: NivStdMod = (/95000,90000,80000,75000,65000,60000,55000,45000,35000,& & 27500,22500,17500/)	INTEGER(KIND=jpim) :: force_profil INTEGER(KIND=jpim), DIMENSION(:),ALLOCATABLE :: ielimine,ilayer INTEGER(KIND=jpim), DIMENSION(12), PARAMETER :: NivStdMod = (/95000,90000,80000,75000,65000,60000,55000,45000,35000,& & 27500,22500,17500/)
none	zatmstd(1:103)=(/ 0._JPRB, 40._JPRB, 80._JPRB, 120._JPRB, 160._JPRB, & & 200._JPRB, 243._JPRB, 287._JPRB, 333._JPRB, 384._JPRB, 440._JPRB, 500._JPRB, & & 566._JPRB, 637._JPRB, 714._JPRB, 798._JPRB, 888._JPRB, 986._JPRB, 1092._JPRB, & & 1206._JPRB, 1328._JPRB, 1459._JPRB, 1600._JPRB, 1750._JPRB, 1910._JPRB, 2080._JPRB, & & 2261._JPRB, 2452._JPRB, 2653._JPRB, 2865._JPRB, 3087._JPRB, 3319._JPRB, 3561._JPRB, & & 3811._JPRB, 4079._JPRB, 4335._JPRB, 4607._JPRB, 4885._JPRB, 5168._JPRB, 5455._JPRB, & & 5745._JPRB, 6038._JPRB, 6331._JPRB, 6624._JPRB, 6915._JPRB, 7205._JPRB, 7494._JPRB, & & 7783._JPRB, 8072._JPRB, 8361._JPRB, 8650._JPRB, 8941._JPRB, 9232._JPRB, 9524._JPRB, & & 9817._JPRB,10110._JPRB,10403._JPRB,10696._JPRB,10987._JPRB,11278._JPRB,11569._JPRB, &	zatmstd(1:103)=(/ 0._JPRB, 40._JPRB, 80._JPRB, 120._JPRB, 160._JPRB, & & 200._JPRB, 243._JPRB, 287._JPRB, 333._JPRB, 384._JPRB, 440._JPRB, 500._JPRB, & & 566._JPRB, 637._JPRB, 714._JPRB, 798._JPRB, 888._JPRB, 986._JPRB, 1092._JPRB, & & 1206._JPRB, 1328._JPRB, 1459._JPRB, 1600._JPRB, 1750._JPRB, 1910._JPRB, 2080._JPRB, & & 2261._JPRB, 2452._JPRB, 2653._JPRB, 2865._JPRB, 3087._JPRB, 3319._JPRB, 3561._JPRB, & & 3811._JPRB, 4079._JPRB, 4335._JPRB, 4607._JPRB, 4885._JPRB, 5168._JPRB, 5455._JPRB, & & 5745._JPRB, 6038._JPRB, 6331._JPRB, 6624._JPRB, 6915._JPRB, 7205._JPRB, 7494._JPRB, & & 7783._JPRB, 8072._JPRB, 8361._JPRB, 8650._JPRB, 8941._JPRB, 9232._JPRB, 9524._JPRB, & & 9817._JPRB,10110._JPRB,10403._JPRB,10696._JPRB,10987._JPRB,11278._JPRB,11569._JPRB, &

	& 11861._JPRB,12157._JPRB,12455._JPRB,12675. _JPRB,13062._JPRB,13371._JPRB,13684._JPRB, & & 14000._JPRB,14319._JPRB,14640._JPRB,14962. _JPRB,15289._JPRB,15633._JPRB,16001._JPRB, & & 16403._JPRB,16846._JPRB,17337._JPRB,17881. _JPRB,18484._JPRB,19147._JPRB,19876._JPRB, & & 20671._JPRB,21535._JPRB,22476._JPRB,23514. _JPRB,24666._JPRB,25945._JPRB,27362._JPRB, & & 28926._JPRB,30642._JPRB,32511._JPRB,34538. _JPRB,36719._JPRB,39048._JPRB,41518._JPRB, & & 44118._JPRB,46836._JPRB,49658._JPRB,52561. _JPRB,55511._JPRB,58495._JPRB,65005._JPRB /)	& 11861._JPRB,12157._JPRB,12455._JPRB,12675._ JPRB,13062._JPRB,13371._JPRB,13684._JPRB, & & 14000._JPRB,14319._JPRB,14640._JPRB,14962._ JPRB,15289._JPRB,15633._JPRB,16001._JPRB, & & 16403._JPRB,16846._JPRB,17337._JPRB,17881._ JPRB,18484._JPRB,19147._JPRB,19876._JPRB, & & 20671._JPRB,21535._JPRB,22476._JPRB,23514._ JPRB,24666._JPRB,25945._JPRB,27362._JPRB, & & 28926._JPRB,30642._JPRB,32511._JPRB,34538._ JPRB,36719._JPRB,39048._JPRB,41518._JPRB, & & 44118._JPRB,46836._JPRB,49658._JPRB,52561._ JPRB,55511._JPRB,58495._JPRB,65005._JPRB /)

IF (ANY(t_date(1:6) == rabsi) .OR. &
& .NOT.
(VerifDate(INT(t_date(1:6))))) THEN
iterr(3) = iterr(3) + 1
DEALLOCATE(Message
%Bodies,STAT=iretalloc)
IF (iretalloc /=0) CALL Abor1("**
ERROR : Temp() cannot deallocate Message
%Bodies(:)")
CYCLE
ENDIF
Message%Head%Date =
t_date(1)*10000+t_date(2)*100+t_date(3)
	IF (t_date(6) == rabsi) t_date(6) =
	0._jprb
	IF (ANY(t_date(1:6) == rabsi)) THEN
	iterr(3) = iterr(3) + 1
	DEALLOCATE(Message
	%Bodies,STAT=iretalloc)
	IF (iretalloc /=0) CALL Abor1("**
	ERROR : Temp() cannot deallocate Message
	%Bodies(:)")
	CYCLE
	ENDIF
	Message%Head%Date =
	t_date(1)*10000+t_date(2)*100+t_date(3)
		IF (t_date(6) == rabsi) t_date(6) =
		0._jprb
		IF (ANY(t_date(1:6) == rabsi)) THEN
		iterr(3) = iterr(3) + 1
		DEALLOCATE(Message
		%Bodies,STAT=iretalloc)
		IF (iretalloc /=0) CALL Abor1("**
		ERROR : Temp() cannot deallocate Message
		%Bodies(:)")
		CYCLE
		ENDIF
		Message%Head%Date =
		t_date(1)*10000+t_date(2)*100+t_date(3)

CASE (7) Message%Head%STypeMessage = SousTypeMessage(1) CASE DEFAULT	CASE (7) Message%Head%STypeMessage = SousTypeMessage(1) CASE DEFAULT Message%Head%STypeMessage = SousTypeMessage(1)	CASE (7) Message%Head%STypeMessage = SousTypeMessage(1) CASE DEFAULT Message%Head%STypeMessage = SousTypeMessage(1)
DEALLOCATE(Message %Bodies,STAT=iretalloc) IF (iretalloc /=0) CALL Abor1("** ERROR : Temp() cannot deallocate Message %Bodies(:)") CYCLE	print *, "*** force au sous- type :",SousTypeMessage(1)	print *, "*** force au sous- type :",SousTypeMessage(1)
! --- recuperation WMO number ou de l'identifiant	! --- informations necessaires pour fcqodb.F90 FcqInfo = Message%Head%TypeMessage * 100000 + & & Message%Head%STypeMessage * 100 + & & Vertical_Unit_Flag SELECT CASE (FcqInfo) ! on ne doit pas rencontrer d'autres cas CASE (503511,503611,503711) Message%Head%FcqInfo = 52 ZLimit = 10000._jprb CASE (513511) Message%Head%FcqInfo = 53 ZLimit = 10000._jprb CASE (603211,603311,613211) Message%Head%FcqInfo = 50 ZLimit = 10000._jprb CASE (603212,603312,613212) Message%Head%FcqInfo = 51 ZLimit = -16179.7_jprb END SELECT ! --- recuperation WMO number ou de l'identifiant	! --- informations necessaires pour fcqodb.F90 FcqInfo = Message%Head%TypeMessage * 100000 + & & Message%Head%STypeMessage * 100 + & & Vertical_Unit_Flag SELECT CASE (FcqInfo) ! on ne doit pas rencontrer d'autres cas CASE (503511,503611,503711) Message%Head%FcqInfo = 52 ZLimit = 10000._jprb CASE (513511) Message%Head%FcqInfo = 53 ZLimit = 10000._jprb CASE (603211,603311,613211) Message%Head%FcqInfo = 50 ZLimit = 10000._jprb CASE (603212,603312,613212) Message%Head%FcqInfo = 51 ZLimit = -16179.7_jprb END SELECT ! --- recuperation WMO number ou de l'identifiant
none	force_profil = 0 ! on force de faire un profil au depart NbValidLevel = Message%Head%NbLevels	force_profil = 0 ! on force de faire un profil au depart NbValidLevel = Message%Head%NbLevels
none	iflag_odb = 0 IF (ANY(NINT(Message %bodies(i+1)%NivPress) .EQ. NivStdMod(:))) iflag_odb = IBSET(iflag_odb,8)	iflag_odb = 0 IF (ANY(NINT(Message %bodies(i+1)%NivPress) .EQ. NivStdMod(:))) iflag_odb = IBSET(iflag_odb,8)
Message%Head%NbLevels = Message %Head%NbLevels - 1	none	none
Message%bodies(i+1)%NivPress = values(j*kel+tconfig%ioffset(1)*i+tconfig %iindice(007009))	Message%bodies(i+1)%NivPress = - (values(j*kel+tconfig%ioffset(1)*i+tconfig %iindice(007009)) * RG) iflag_odb = 0	Message%bodies(i+1)%NivPress = - (values(j*kel+tconfig%ioffset(1)*i+tconfig %iindice(007009)) * RG) iflag_odb = 0
IF (iflag_omm /= 262143) THEN	IF (iflag_omm /= 262143) THEN	IF (iflag_omm /= 262143) THEN

iflag_odb = 0	! iflag_odb = 0	! iflag_odb = 0

IF (IBITS(iflag_omm,11,1) == 1) iflag_odb = IBSET(iflag_odb,4) ! significant wind level
iflag_odb = IBSET(iflag_odb,Vertical_Unit_Flag) ! vertical coordinate in Pa or m.
Message%Bodies(i+1)%Flag = iflag_odb
	IF (IBITS(iflag_omm,11,1) == 1) iflag_odb = IBSET(iflag_odb,4) ! significant wind level
	iflag_odb = IBSET(iflag_odb,Vertical_Unit_Flag) ! vertical coordinate in Pa or m.
	IF (IBITS(iflag_odb,8,1) == 1 .AND. Message%bodies(i+1)%NivPress < ZLimit) THEN
	iflag_odb = IBSET(iflag_odb,14) ! Part C
	ENDIF
	IF (IBITS(iflag_odb,8,1) == 0 .AND. Message%bodies(i+1)%NivPress < ZLimit) THEN
	iflag_odb = IBSET(iflag_odb,13) ! Part D
	iflag_odb = IBSET(iflag_odb,14) ! Part D
	ENDIF
	IF (IBITS(iflag_odb,8,1) == 0 .AND. Message%bodies(i+1)%NivPress >= ZLimit) THEN
	iflag_odb = IBSET(iflag_odb,13) ! Part B
	ENDIF
	IF (Vertical_Unit_Flag == 12) Message%bodies(i+1)%NivPress = ABS(Message%bodies(i+1)%NivPress)
	Message%Bodies(i+1)%Flag = iflag_odb
		IF (IBITS(iflag_omm,11,1) == 1) iflag_odb = IBSET(iflag_odb,4) !significant wind level
		iflag_odb = IBSET(iflag_odb,Vertical_Unit_Flag) ! vertical coordinate in Pa or m.
		IF (IBITS(iflag_odb,8,1) == 1 .AND. Message%bodies(i+1)%NivPress < ZLimit) THEN
		iflag_odb = IBSET(iflag_odb,14) ! Part C
		ENDIF
		IF (IBITS(iflag_odb,8,1) == 0 .AND. Message%bodies(i+1)%NivPress < ZLimit) THEN
		iflag_odb = IBSET(iflag_odb,13) ! Part D
		iflag_odb = IBSET(iflag_odb,14) ! Part D
		ENDIF
		IF (IBITS(iflag_odb,8,1) == 0 .AND. Message%bodies(i+1)%NivPress >= ZLimit) THEN
		iflag_odb = IBSET(iflag_odb,13) ! Part B
		ENDIF
		IF (Vertical_Unit_Flag == 12) Message%bodies(i+1)%NivPress = ABS(Message%bodies(i+1)%NivPress)
		Message%Bodies(i+1)%Flag = iflag_odb

none	force_profil = 1 ! on autorise un traitement type 'avion'	force_profil = 1 ! on autorise un traitement type 'avion'

	Message%Head%NbLevels = NbValidLevel
	
	!FGFG type de temp 0=TAC, 1=std, 2=HR
	IF (Message%Head%ammendement < 50 .AND. Message%Head%NbLevels > NbMinLevelHr) Message%Head%FcqInfo = Message%Head%FcqInfo+200
	IF (Message%Head%ammendement < 50 .AND. Message%Head%NbLevels <= NbMinLevelHr) Message%Head%FcqInfo = Message%Head%FcqInfo+100
	
	!FGFG on supprime tout ce qui n'est pas avec delta lat/lon/time pour les temp
	SELECT CASE (Message%Head%TypeMessage*10+force_profil)
	CASE (50)
	IF (ElimTemp0) THEN
	DEALLOCATE(Message%Bodies,STAT=iretalloc)
	IF (iretalloc /=0) CALL Abor1("** ERROR : Temp() cannot deallocate Message%Bodies(:)")
	CYCLE
	ENDIF
	CASE (60)
	IF (ElimPilot0) THEN
	DEALLOCATE(Message%Bodies,STAT=iretalloc)
	IF (iretalloc /=0) CALL Abor1("** ERROR : Temp() cannot deallocate Message%Bodies(:)")
	CYCLE
	ENDIF
	END SELECT
	
	select case (Message%Head%TypeMessage)
	case (5)
	if (Message%Head%NbLevels > NbMinLevelHr) then
	allocate(ielimine(Message%Head%NbLevels),stat=iretalloc)
	allocate(ilayer(Message%Head%NbLevels),stat=iretalloc)
	ielimine(:)=0
	do i1 = 1, Message%Head%NbLevels
	ilayer(i1)=103
	do iz=2,103
	if(Message%bodies(i1)%Geopo < (zatmstd(iz)*9.81_JPRB)) then
	ilayer(i1)=iz-1
	exit
	endif
	enddo
	enddo
	do i1 = 2, Message%Head%NbLevels
	if (IBITS(NINT(Message%Bodies(i1)%Flag),8,1) == 1) cycle
	zz1=Message%bodies(i1)%Geopo
	zdeltaz=(zatmstd(ilayer(i1)+1)-
		Message%Head%NbLevels = NbValidLevel
		
		!FGFG type de temp 0=TAC, 1=std, 2=HR
		IF (Message%Head%ammendement < 50 .AND. Message%Head%NbLevels > NbMinLevelHr) Message%Head%FcqInfo = Message%Head%FcqInfo+200
		IF (Message%Head%ammendement < 50 .AND. Message%Head%NbLevels <= NbMinLevelHr) Message%Head%FcqInfo = Message%Head%FcqInfo+100
		
		!FGFG on supprime tout ce qui n'est pas avec delta lat/lon/time pour les temp
		SELECT CASE (Message%Head%TypeMessage*10+force_profil)
		CASE (50)
		IF (ElimTemp0) THEN
		DEALLOCATE(Message%Bodies,STAT=iretalloc)
		IF (iretalloc /=0) CALL Abor1("** ERROR : Temp() cannot deallocate Message%Bodies(:)")
		CYCLE
		ENDIF
		CASE (60)
		IF (ElimPilot0) THEN
		DEALLOCATE(Message%Bodies,STAT=iretalloc)
		IF (iretalloc /=0) CALL Abor1("** ERROR : Temp() cannot deallocate Message%Bodies(:)")
		CYCLE
		ENDIF
		END SELECT
		
		select case (Message%Head%TypeMessage)
		case (5)
		if (Message%Head%NbLevels > NbMinLevelHr) then
		allocate(ielimine(Message%Head%NbLevels),stat=iretalloc)
		allocate(ilayer(Message%Head%NbLevels),stat=iretalloc)
		ielimine(:)=0
		do i1 = 1, Message%Head%NbLevels
		ilayer(i1)=103
		do iz=2,103
		if(Message%bodies(i1)%Geopo < (zatmstd(iz)*9.81_JPRB)) then
		ilayer(i1)=iz-1
		exit
		endif
		enddo
		enddo
		do i1 = 2, Message%Head%NbLevels
		if (IBITS(NINT(Message%Bodies(i1)%Flag),8,1) == 1) cycle
		zz1=Message%bodies(i1)%Geopo
		zdeltaz=(zatmstd(ilayer(i1)+1)-

	zatmstd(ilayer(i1)))*9.81_JPRB/
	(real(NFREQVERT_TPHR)/51._JPRB)
	do i2 = 1,i1-1
	if (ielimine(i2) == 0) then
	zz2=Message%bodies(i2)%Geopo
	if (abs(zz2-zz1) < zdeltaz)
	then
	ielimine(i1)=1
	Message%bodies(i1)%T2m
	= rabso
	Message%bodies(i1)%Td
	= rabso
	Message%bodies(i1)%NivPress
	= rabso
	Message%bodies(i1)%Geopo
	= rabso
	Message%bodies(i1)%dd
	= rabso
	Message%bodies(i1)%ff
	= rabso
	Message%bodies(i1)%Flag
	= rabso
	exit
	endif
	endif
	enddo
	enddo
	do i1 = 2, Message%Head%NbLevels
	if (IBITS(NINT(Message
	%Bodies(i1)%Flag),8,1) == 1) cycle
	if (ielimine(i1) == 0) then
	zz1=Message%bodies(i1)%Geopo
	zdeltaz=(zatmstd(ilayer(i1)+1)-
	zatmstd(ilayer(i1)))*9.81_JPRB/
	(real(NFREQVERT_TPHR)/102._JPRB)
	do i2 = 1,i1-1
	if ((ielimine(i2) == 0) .AND.
	(ilayer(i2) == ilayer(i1))) then
	zz2=Message%bodies(i2)%Geopo
	if (abs(zz2-zz1) <
	zdeltaz) then
	ielimine(i1)=1
	Message%bodies(i1)%T2m
	= rabso
	Message%bodies(i1)%Td
	= rabso
	Message
	%bodies(i1)%NivPress = rabso
	Message%bodies(i1)%Geopo
	= rabso
	Message%bodies(i1)%dd
	= rabso
	Message%bodies(i1)%ff
	= rabso
	Message%bodies(i1)%Flag
	= rabso
	exit
	endif
	endif
	enddo
	endif
	enddo
	
	write(0,*) ' ON CONSERVE ',Message
	%Head%NbLevels-sum(ielimine),' NIVEAUX SUR
	',Message%Head%NbLevels
	deallocate(ielimine,stat=iretalloc)
	deallocate(ilayer,stat=iretalloc)
	IF (iretalloc /=0) CALL Abor1("**
	ERROR : Temp() cannot deallocate
	ielimine(:)")
	endif
	end select
		zatmstd(ilayer(i1)))*9.81_JPRB/
		(real(NFREQVERT_TPHR)/51._JPRB)
		do i2 = 1,i1-1
		if (ielimine(i2) == 0) then
		zz2=Message%bodies(i2)%Geopo
		if (abs(zz2-zz1) < zdeltaz)
		then
		ielimine(i1)=1
		Message%bodies(i1)%T2m
		= rabso
		Message%bodies(i1)%Td
		= rabso
		Message%bodies(i1)%NivPress
		= rabso
		Message%bodies(i1)%Geopo
		= rabso
		Message%bodies(i1)%dd
		= rabso
		Message%bodies(i1)%ff
		= rabso
		Message%bodies(i1)%Flag
		= rabso
		exit
		endif
		endif
		enddo
		enddo
		do i1 = 2, Message%Head%NbLevels
		if (IBITS(NINT(Message
		%Bodies(i1)%Flag),8,1) == 1) cycle
		if (ielimine(i1) == 0) then
		zz1=Message%bodies(i1)%Geopo
		zdeltaz=(zatmstd(ilayer(i1)+1)-
		zatmstd(ilayer(i1)))*9.81_JPRB/
		(real(NFREQVERT_TPHR)/102._JPRB)
		do i2 = 1,i1-1
		if ((ielimine(i2) == 0) .AND.
		(ilayer(i2) == ilayer(i1))) then
		zz2=Message%bodies(i2)%Geopo
		if (abs(zz2-zz1) < zdeltaz)
		then
		ielimine(i1)=1
		Message%bodies(i1)%T2m
		= rabso
		Message%bodies(i1)%Td
		= rabso
		Message%bodies(i1)%NivPress
		= rabso
		Message%bodies(i1)%Geopo
		= rabso
		Message%bodies(i1)%dd
		= rabso
		Message%bodies(i1)%ff
		= rabso
		Message%bodies(i1)%Flag
		= rabso
		exit
		endif
		endif
		enddo
		endif
		enddo
		
		write(0,*) ' ON CONSERVE ',Message
		%Head%NbLevels-sum(ielimine),' NIVEAUX SUR
		',Message%Head%NbLevels
		deallocate(ielimine,stat=iretalloc)
		deallocate(ilayer,stat=iretalloc)
		IF (iretalloc /=0) CALL Abor1("**
		ERROR : Temp() cannot deallocate
		ielimine(:)")
		endif
		end select

CALL Post_Radiosondage(kobs,kw,Message,t_date,i terr)	none	none
! Subroutine/Function | Post_Temp	! Subroutine/Function | Post_Radiodondage1	! Subroutine/Function | Post_Radiodondage1
none	SUBROUTINE Post_Radiosondage2 (FULL)	SUBROUTINE Post_Radiosondage2 (FULL)
none	SUBROUTINE Post_Radiosondage3 (FULL)	SUBROUTINE Post_Radiosondage3 (FULL)

*traitement particulier des radomes francais

Appendix C: BatorODB working environment in Portugal (UBUNTU)

The following working tools have been used to check the messages contents:

- bufr_decode_all (ECMWF package bufrdc_000405), used locally to decode the BUFR messages
- decode_bufr (M-F package auxlibs_installer.2.6), used locally to decode the BUFR messages
- odbsql, used at ECMWF to read the ODB contents.

Appendix D: BatorODB working environment in Portugal (AIX)

In order to repeat the exercise in Portugal, it was also necessary to prepare the compilation environment under development mode. In this way, the following steps have been done:

(i) to set up gmkpack environment on a local user to compile separately the merged pieces of code;
(ii) in order to recreate the BATOR executable, the following sources have been copied from the export version:

bator_datetime_mod.F90
bator_decodbufr_mod.F90
bator_module.F90

to the following directory (on the local, Portuguese machine):

/u/monteiro/pack/cy38t1_bator/src/local/odb/pandor/module

In order to re-create the compilation script, ics_bator, that is at the directory

/u/monteiro/pack/cy38t1_bator

the command genpack has been given and its content repeated at the command line.

For each compilation trial, a new executable is created at the directory:

/u/monteiro/pack/cy38t1_bator/bin