Apêndice A

Programa de Navegação das Imagens Obtidas pelo MSG-1

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Introduction:
! =============
! The Program "MSG_navigation.f90" is an example code provided to give
! the users guidance for a possible implementation of the equations
! given in the LRIT/HRIT Global Specification [1] to navigate MSG
! (METEOSAT 8 onwards) data, i.e. to link the pixel coordinates column
! and line to the corresponding geograhical coordinates latitude and
! longitude.
!
! Users should take note, however, that it does NOT provide software
! for reading MSG data either in LRIT/HRIT, in native or any other
! format and that EUMETSAT cannot guarantee the accuracy of this
! software. The software is for use with MSG data only and will not
! work in the given implementation for Meteosat first generation data.
!
! Two functions/subroutines are provided:
!   pixcoord2geocoord: for conversion of column/line into lat./long.
!   geocoord2pixcoord: for conversion of lat./long. into column/line
!
! The main routine gives an example how to utilize these two subroutines by
! reading a value for column and line at the start of the program on the
! command line and convert these value into the cooresponding
! geographical coordinates and back again. The results are then printed
! out on the screen.
!
! To Compile the program use for example:
!
! COMMAND PROMPT: f90 MSG_navigation.c -o MSG_navigation
!
! Run the program by typing
```

```
!
! COMMAND PROMPT: ./MSG_navigation <COLUMS> <ROWS>
!
! ---------------------------------------------------------------------
!
! NOTE: Please be aware, that the program assumes the MSG image is
! ordered in the operational scanning direction which means from south
! to north and from east to west. With that the VIS/IR channels contains
! of 3712 x 3712 pixels, start to count on the most southern line and the
! most eastern column with pixel 1,1.
!
! NOTE on CFAC/LFAC and COFF/LOFF
! The parameters CFAC/LFAC and COFF/LOFF are the scaling
! coefficients provided by the navigation record and used by the scaling
! function given in Ref [1], page 28.
!
! COFF/LOFF are the offsets for column and line which are basically 1856
! and 1856 for the VIS/IR channels and refer to the middle of the image
! (centre pixel). The values regarding the High Resolution Visible Channel
! (HRVis) will be made available in a later issue of this software.
!
! CFAC/LFAC are responsible for the image "spread" in the NS and EW
! directions. They are calculated as follows:
! CFAC = LFAC = 2^16 / delta
! with
! delta = 83.843 micro Radian (size of on VIS/IR MSG pixel)
!
! CFAC     = LFAC     =   781648343.404  rad^-1 for VIS/IR
!
! which should be rounded to the nearest integer as stated in Ref [1].
!
! CFAC     = LFAC     =   781648343  rad^-1 for VIS/IR
!
! the sign of CFAC/LFAC gives the orientation of the image.
! Negative sign give data scanned from south to north as in the
! operational scanning. Positive sign vice versa.
!
! The terms "line" and "row" are used interchangeable.
!
! PLEASE NOTE that the values of CFAC/LFAC which are given in the
! Header of the LRIT/HRIT Level 1.5 Data (see [2]) are actually in
! Degrees and should be converted in Radians for use with these
! routines (see example and values above).
!
! The other parameters are given in Ref [1].
!
```

```
! Further information may be found in either Ref [1], Ref [2] or Ref [3] or
! on the Eumetsat website http://www.eumetsat.de/ .
!
!  REFERENCE:
!  [1] LRIT/HRIT Global Specification
!      (CGMS 03, Issue 2.6, 12.08.1999)
!      for the parameters used in the program.
!  [2] MSG Ground Segment LRIT/HRIT Mission Specific
!      Implementation, EUMETSAT Document,
!      (EUM/MSG/SPE/057, Issue 5, 4. February 2005).
!  [3] MSG Level 1.5 Image Data Format Description
!      (EUM/PS-MSG/ICD/04/0730, Issue 3, 4. February 2005).
!
!
! Please email the User Service (via
! //http://www.eumetsat.de/feedback/form_ops.html)
! if you have any questions regarding this software.
!
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


MODULE GLOBAL
  ! define global parameters used in the routines as given in Ref. [1].
  ! to distinguish between them and other variables they are written
  ! in CAPITAL LETTERS.

  REAL, PARAMETER :: PI=3.14159265359

  REAL, PARAMETER ::  SAT_HEIGHT= 42164.0
  REAL, PARAMETER ::  R_EQ = 6378.169
  REAL, PARAMETER ::  R_POL= 6356.5838
  REAL, PARAMETER ::  SUB_LON = 0.0     ! Longitude of Sub-Satellite Point

  INTEGER, PARAMETER ::  CFAC = -781651420
  INTEGER, PARAMETER ::  LFAC = -781651420

  INTEGER, PARAMETER ::  COFF = 1856
  INTEGER, PARAMETER ::  LOFF = 1856

END MODULE GLOBAL

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! The following program gives an example how to incorporate the
! functions "geocoord2pixcoord" and "pixcoord2geocoord"
```

```
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


PROGRAM navi

  USE GLOBAL

  IMPLICIT NONE

! command line variables
  CHARACTER *100 BUFFER
  integer k

! working variables
  INTEGER :: column, row, line, col
  INTEGER :: c, l
  INTEGER :: ccc, lll

  DOUBLE PRECISION :: d_lat, d_lon
  DOUBLE PRECISION :: latitude
  DOUBLE PRECISION :: longitude

  k = iargc()

  ! command line
  if ( k /= 2 ) then
     print *, " call of program "
     print *, " ./MSG_navigation <COLUMS> <ROWS> "
     print *, " EXIT! "
     stop
  end if

  call getarg(1,BUFFER)
  read(buffer,*) column
  call getarg(2,BUFFER)
  read(buffer,*) row


  ! now start working
  c=column
  l=row


  ! first convert the pixel column and row to the appropriate
  ! longitue/latitude pair
```

```
  call pixcoord2geocoord(c, l, COFF, LOFF, latitude, longitude)


  ! use the result from "pixcoord2geocoord" to convert back into
  ! the pixel column and row again

  call geocoord2pixcoord(latitude, longitude, COFF, LOFF, ccc, lll)

  ! print out results
  print *, "c=",c," l=",l," lat=",latitude," lon=",longitude, " col=",ccc,
*        "row=",lll

END PROGRAM navi

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! subroutine pixcoord2geocoord
!
!  PURPOSE:
!  return the geograhic latitude and longitude of an MSG image
!    for a given pair of latitude/longitude.
!    (based on the formulas given in Ref. [1])
!
!
!  DEPENDENCIES:
!    none
!
!
!  REFERENCE:
!  [1] LRIT/HRIT Global Specification
!      (CGMS 03, Issue 2.6, 12.08.1999)
!      for the parameters used in the program
!  [2] MSG Ground Segment LRIT/HRIT Mission Specific
!      Implementation, EUMETSAT Document,
!      (EUM/MSG/SPE/057, Issue 5, 4. February 2005)
!
!  MODIFICATION HISTORY:
!    Version 1.0
!    17.01.2005 written by Knut Dammann
!    (c) EUMETSAT 2005
!
!
!  INPUT:
!    row   (int) row-value of the pixel
!    colum (int) columb-value of the pixel
!    ccoff (int) coefficient of the scalling function
```

```
!                (see page 28, Ref [1])
!                NOTE: since Fortran cannot distinguish between
!                      upper case and lower case letters the name
!                      "ccoff" is used rather than "COFF" to distinguish
!                      between them.
!    lloff (int) coefficient of the scalling function
!                (see page 28, Ref [1])
!                NOTE: since Fortran cannot distinguish between
!                      upper case and lower case letters the name
!                      "ccoff" is used rather than "COFF" to distinguish
!                      between them.
!
!  OUTPUT:
!    latitude (double) geographic Latitude of the wanted pixel [Degrees]
!    longitude (double) geographic Longitude of the wanted pixel [Degrees]
!
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

SUBROUTINE pixcoord2geocoord( column,  row,  ccoff,  lloff, latitude,
*        longitude)

  USE GLOBAL

  IMPLICIT NONE

  INTEGER, INTENT (IN) :: column, row, ccoff, lloff
  DOUBLE PRECISION , INTENT (OUT) :: latitude, longitude


  DOUBLE PRECISION :: s1, s2, s3, sn, sd, sxy
  DOUBLE PRECISION :: x, y, xx, yy
  DOUBLE PRECISION :: longi, lati
  DOUBLE PRECISION :: lat
  DOUBLE PRECISION :: lon


  DOUBLE PRECISION :: a, b, sa

  INTEGER ::  c, l


  c=column
  l=row

  !  calculate viewing angle of the satellite by use of the equation
```

```
!  on page 28, Ref [1].

x = (2**16. * ( c - ccoff) ) / real(CFAC)
y = (2**16. * ( l - lloff) ) / real(LFAC)


!  now calculate the inverse projection using equations on page 25, Ref. [1]

!  first check for visibility, whether the pixel is located on the earth
!  surface or in space.
!  To do this calculate the argument to sqrt of "sd", which is named "sa".
!  If it is negative then the sqrt will return NaN and the pixel will be
!  located in space, otherwise all is fine and the pixel is located on the
!  earth surface.

sa =  (SAT_HEIGHT * cos(x) * cos(y) )**2 - (cos(y)*cos(y) + 1.006803 *
    *        sin(y)*sin(y)) * 1737121856.


! now calculate the rest of the formulas using eq. on page 25 Ref [1]

sd = sqrt( (SAT_HEIGHT * cos(x) * cos(y) )**2 - (cos(y)*cos(y) + 1.006803 *
    *        sin(y)*sin(y)) * 1737121856. )
sn = (SAT_HEIGHT * cos(x) * cos(y) - sd) / ( cos(y)*cos(y) + 1.006803 *
    *        sin(y)*sin(y) )

s1 = SAT_HEIGHT - sn * cos(x) * cos(y)
s2 = sn * sin(x) * cos(y)
s3 = -sn * sin(y)

sxy = sqrt( s1*s1 + s2*s2 )


! using the previous calculations now the inverse projection can be
! calculated, which means calculating the lat./long. from the pixel
! row and column by equations on page 25, Ref [1].

! take care if the pixel is in space, that an error code will be returned

if ( sa <= 0. ) then
   latitude = -999.999
   longitude = -999.999
else
   longi = atan(s2/s1 + SUB_LON)
   lati  = atan((1.006803*s3)/sxy)
end if
```

```
  ! convert from radians into degrees
  latitude = lati*180./PI
  longitude = longi*180./PI


END SUBROUTINE pixcoord2geocoord


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! subroutine geocoord2pixcoord
!
! PURPOSE:
!   return the pixel column and line of an MSG image
!   for a given pair of geographic latitude/longitude.
!   (based on the formulas given in Ref. [1])
!
!
! DEPENDENCIES:
!   up to now none
!
!
! REFERENCE:
! [1] LRIT/HRIT Global Specification
!     (CGMS 03, Issue 2.6, 12.08.1999)
!     for the parameters used in the program.
! [2] MSG Ground Segment LRIT/HRIT Mission Specific
!     Implementation, EUMETSAT Document,
!     (EUM/MSG/SPE/057, Issue 5, 4. February 2005).
!
!
! MODIFICATION HISTORY:
!   Version 1.0
!   17.01.2005 written by Knut Dammann
!    (c) EUMETSAT 2005
!
!
! INPUT:
!   latitude  (double) geographic Latitude of a point [Degrees]
!   longitude (double) geographic Longitude of a point [Degrees]
!   ccoff (int)        coefficient of the scalling function
!                      (see page 28, Ref [1])
!                      NOTE: since Fortran cannot distinguish between
!                            upper case and lower case letters the name
!                            "ccoff" is used rather than "COFF"
!                            to distinguish between them.
```

```
!
!   lloff (int)        coefficient of the scalling function
!                      (see page 28, Ref [1])
!                      NOTE: since Fortran cannot distinguish between
!                            upper case and lower case letters the name
!                            "lloff" is used rather than "LOFF"
!                            to distinguish between them.
!
!
! OUTPUT:
!   row    (int)       row-value of the pixel
!   column (int)       column-value of the pixel
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

SUBROUTINE geocoord2pixcoord( latitude,  longitude,  ccoff,  lloff,  column,
           *        row)

  USE GLOBAL

  IMPLICIT NONE

  DOUBLE PRECISION, INTENT (IN) :: latitude, longitude
  INTEGER, INTENT (IN)  :: ccoff, lloff
  INTEGER, INTENT (OUT) :: column, row

  INTEGER :: c=0, l=0
  INTEGER :: ccc=0, lll=0
  INTEGER :: x=0, y=0

  DOUBLE PRECISION :: lati, longi
  DOUBLE PRECISION :: c_lat
  DOUBLE PRECISION :: lat
  DOUBLE PRECISION :: lon
  DOUBLE PRECISION :: r1, r2, r3, rn, re, rl
  DOUBLE PRECISION :: xx, yy, sa
  DOUBLE PRECISION :: cc, ll
  DOUBLE PRECISION :: dotprod


  lati= latitude
  longi= longitude

  ! check if the values are sane, otherwise return error value
  if (lati < -90.0 .or. lati > 90.0 .or. longi < -180.0 .or.
*       longi > 180.0 ) then
```

```
      row = -999
      column = -999
      return
   end if

   ! convert them to radians
   lat = lati*PI / 180.
   lon = longi *PI / 180.


   ! calculate the geocentric latitude from the
   ! geographic one using equations on page 24, Ref. [1]

   c_lat = atan ( (0.993243*(sin(lat)/cos(lat)) ))


   ! using c_lat calculate the length from the earth
   ! centre to the surface of the earth ellipsoid
   ! equations on page 24, Ref [1]

   re = R_POL / sqrt( (1 - 0.00675701 * cos(c_lat) * cos(c_lat) ) )


   ! calculate the forward projection using equations on page 24, Ref. [1]

   rl = re
   r1 = SAT_HEIGHT - rl * cos(c_lat) * cos(lon - SUB_LON)
   r2 = - rl *  cos(c_lat) * sin(lon - SUB_LON)
   r3 = rl * sin(c_lat)
   rn = sqrt( r1*r1 + r2*r2 +r3*r3 )

   ! check for visibility, whether the point on the earth given by the
   ! latitude/longitude pair is visible from the satellte or not. This
   ! is given by the dot product between the vectors of:
   ! 1) the point to the spacecraft,
   ! 2) the point to the centre of the earth.
   ! If the dot product is positive the point is visible otherwise it
   ! is invisible.

   dotprod = r1*(rl * cos(c_lat) * cos(lon - SUB_LON)) - r2*r2 - r3*r3*
 *    ((r_EQ/R_POL)**2)

   if (dotprod <= 0. ) then
      column = -999
      row = -999
      return
```

```
  end if

  ! the forward projection is x and y

  xx = atan( (-r2/r1) )
  yy = asin( (-r3/rn) )


  ! convert to pixel column and row using the scaling functions on
  ! page 28, Ref. [1]. And finding nearest integer value for them.

  cc = ccoff + xx *  2**(-16.) * real(CFAC)
  ll = lloff + yy *  2**(-16.) * real(LFAC)


  ccc=nint(cc)
  lll=nint(ll)

  column = ccc
  row = lll



END SUBROUTINE geocoord2pixcoord
```