PROGRAM BUFR\_TRANAMSUA

C$$$ MAIN PROGRAM DOCUMENTATION BLOCK

C

C MAIN PROGRAM: BUFR\_TRANAMSUA

C PRGMMR: KEYSER ORG: NP22 DATE: 2005-06-21

C

C ABSTRACT: Read raw AMSU-A 1B format file, decode, write selected

C observations to output file. Tb and Ta reports are written to

C unique BUFR files. Tb reports are also written to an IEEE file.

C

C PROGRAM HISTORY LOG:

C 1998-06-15 Treadon -- Original author

C 2000-09-06 Woollen -- Added second output in BUFR

C 2000-11-20 Treadon -- Generalized to allow for different antenna

C to brightness temperature conversion constants for NOAA-15 vs.

C NOAA-16; Changed to properly relabel NOAA-16 satellite id from

C 2 to 16 to be consistent with the convention followed in the

C global and regional analysis systems; Added error handling when

C no output is created so that subsequent TRANJB's are skipped

C 2002-02-11 Woollen -- Modifications and corrections to output BUFR

C dataset: "SAID" (0-01-007) corrected to proper WMO Code Table

C value (was 14 for NOAA-14, etc.), "SIID" (0-02-019) repl.

C "SIDU" (0-02-021) which didn't seem to be correct, "HMSL"

C (0-07-002) corrected to proper units of meters (was being

C stored in km), "LSQL" (0-08-012) corrected to proper WMO Code

C Table value (0-land/1-sea) (was backwards), "TMBR" (0-12-163)

C corrected to proper units of K (was being stored as K +

C 273.15), channel 20 "TMBR" set to missing for HIRS-2 and HIRS-3

C types

C 2002-07-08 Keyser -- Accounts for NOAA-17 (converts NESDIS sat.

C no. from 6 to 17); Sets default values for Ta to Tb

C coefficients which are used when coefficient file is empty or

C missing (before program failed in this situation)

C 2002-10-23 Treadon -- Use lbyte instead of mbyte for unpacking

C counts

C 2004-01-23 Keyser -- Based on new namelist switch "compress", now

C has option to write compressed BUFR messages using WRITCP

C instead of WRITSB (removes the need for the downstream program

C BUFR\_COMPRESS)

C 2005-04-29 Keyser -- Added processing of Ta reports to new, unique

C BUFR file in unit 53

C 2005-06-21 Keyser -- Modified to handle processing of NOAA-18 data

C

C USAGE:

C

C INPUT FILES:

C UNIT 05 - Standard input (namelist "input")

C UNIT 11 - Binary file containing raw 1B AMSU-A data

C UNIT 12 - BUFR mnemonic table

C UNIT 21 - Binary file containing Ta to Tb coefficients

C UNIT 41 - Binary file containing topography information

C used by function LANSEA

C

C \*\*\*NOTE\*\*\*

C Function LANSEA assumes this information is in

C a file named 'lowtopog.dat' which is local to

C the working directory.

C

C OUTPUT FILES:

C UNIT 06 - Printout

C UNIT 51 - Binary (IEEE) file containing decoded AMSU-A Tb

C data

C UNIT 52 - BUFR file containing AMSU-A Tb data

C UNIT 53 - BUFR file containing AMSU-A Ta data

C

C SUBPROGRAMS CALLED:

C UNIQUE: - AMSUA CHARS DATTIM ICHARS LANSEA LBIT

C MBYTE LBYTE XFLOAT BUFR1B

C SYSTEM: - SYSTEM

C LIBRARY:

C W3LIB - W3TAGB W3TAGE ERREXIT W3FS26

C BUFRLIB - OPENBF CLOSBF OPENMB WRITSB WRITCP UFBSEQ

C MESGBC

C

C

C EXIT STATES

C 0 = No errors detected

C 1 = Data type id decoded from header is not for AMSU-A

C 2 = Problem reading Ta to Tb coefficient file

C 3 = Problem reading header record of 1b AMSU-A file

C 6 = Unknown satellite id

C 7 = Unknown satellite instrument

C

C REMARKS:

C Switches read in Namelist INPUT:

C INFILE - Path to input 1B data file

C OUTFILE - Path to output IEEE file

C COMPRESS - BUFR compression switch (YES or NO)

C COEFILE - Path to input coefficient file

C PROCESS\_Tb - Process brightness temps into BUFR and IEEE files?

C PROCESS\_Ta - Process antenna temps into BUFR files?

C

C####################################################################

C NOTE: This program is designed to process BOTH Ta and Tb into

C separate BUFR files in a single run. However, BUFRLIB

C routine WRITCP (which writes COMPRESSED messages) cannot

C operate on two output files at the same time. If it is

C ever modified to do so (like WRITSB which writes uncompressed

C BUFR messages), then the code to ready to handle this. In

C the meantime, this code must be executed twice to process

C both Ta and Tb (once with process\_Ta=YES and process\_Tb=NO

C and again with process\_Ta=NO and process\_Tb=YES).

C####################################################################

C

C ATTRIBUTES:

C LANGUAGE: FORTRAN 90

C MACHINE: NCEP-CCS

C

C$$$

C Declare namelist variables and namelist

C ---------------------------------------

integer stdout

character\*80 infile,outfile,coefile

character\*8 compress,process\_Tb,process\_Ta

namelist /input/ infile,outfile,coefile,compress,process\_Tb,

x process\_Ta

common/switches/compress,process\_Tb,process\_Ta

C Set I/O unit numbers

C --------------------

data lunam, stdout / 5, 6 /

data lunin, luncof / 11, 21 /

data lunout / 51 /

call w3tagb('BUFR\_TRANAMSUA',2005,0172,0068,'NP22')

print \*

print \*, 'WELCOME TO BUFR\_TRANAMSUA - Version 06/21/2005'

print \*

C Get Namelist input

C ------------------

read(lunam,input)

write(stdout,\*)'namelist input below'

write(stdout,input)

C Open unit to output IEEE file

C -----------------------------

open(lunout,file=outfile,form='unformatted')

C Read/decode/output data records scan by scan

C --------------------------------------------

call amsua(lunin,infile,luncof,coefile,lunout)

call w3tage('BUFR\_TRANAMSUA')

stop

end

cfpp$ expand(lbyte,mbyte,xfloat,lansea,dattim)

SUBROUTINE AMSUA(LUNIN,RAWAMSU,LUNCOF,COEFILE,LUNOUT)

C$$$ SUBPROGRAM DOCUMENTATION BLOCK

C

C SUBPROGRAM: AMSUA

C PRGMMR: D. A. KEYSER ORG: NP22 DATE: 2005-06-21

C

C ABSTRACT: Read raw AMSU-A 1B format file, decode, write selected

C observations to output file. Tb and Ta reports are written to

C unique BUFR files. Tb reports are also written to an IEEE file.

C

C PROGRAM HISTORY LOG:

C 1998-06-15 Treadon -- Original author

C 2002-07-08 Keyser -- Accounts for NOAA-17 (converts NESDIS sat.

C no. from 6 to 17)

C 2004-05-03 Keyser -- Added processing of Ta reports to new, unique

C BUFR file in UNIT 53

C 2005-06-21 Keyser -- Modified to handle processing of NOAA-18 data

C

C USAGE: CALL AMSUA(LUNIN,RAWAMSU,LUNCOF,COEFILE,LUNOUT)

C INPUT ARGUMENT LIST:

C LUNIN - Unit connected to raw 1B AMSU-A data file

C RAWAMSU - Name of raw 1B AMSU-A data file

C LUNCOF - Unit connected to Ta to Tb conversion coefficients

C LUNOUT - Unit connected to output binary (IEEE) file

C

C INPUT FILES:

C UNIT 12 - BUFR mnemonic table

C UNIT 41 - Binary file containing topography information

C used by function LANSEA

C UNIT LUNIN - Binary file containing raw 1B AMSU-A data

C UNIT LUNCOF - Binary file containing Ta to Tb coefficients

C

C \*\*\*NOTE\*\*\*

C Function LANSEA assumes this information is in

C a file named 'lowtopog.dat' which is local to

C the working directory.

C

C OUTPUT FILES:

C UNIT 06 - Printout

C UNIT 52 - BUFR file containing AMSU-A Tb data

C UNIT 53 - BUFR file containing AMSU-A Ta data

C UNIT LUNOUT - Binary (IEEE) file containing decoded AMSU-A Tb

C data

C

C REMARKS:

C

C ATTRIBUTES:

C LANGUAGE: FORTRAN 90

C MACHINE: NCEP-CCS

C

C$$$

C Include machine dependent parameters

C ------------------------------------

include 'rfac.inc'

C Declare/set parameters:

C NBYTE1 = Total number of bytes (2560) in AMSU-A data record

C NBYTE4 = Number of 4-byte words in NBYTE1 bytes (2560/4=640)

C NSET = Number of topography datasets for function LANSEA3

C (not currently used)

C NID,NND = Number of arguments in arrays in dattim call

C EPS = A "small" number

C MCH = Number of channels

C MPOS = Number of spots (positions) on a scan line

integer,parameter::real\_32=selected\_real\_kind(6,37)

integer,parameter::real\_64=selected\_real\_kind(15,307)

real(real\_64) eps

parameter (nbyte1=2560,nbyte4=nbyte1/4)

parameter (nset=3)

parameter (nid=3,nnd=6)

parameter (eps=1.d-12)

parameter (mch=15)

parameter (mpos=30)

C Set parameters for structure of output data file

C ------------------------------------------------

parameter (nreal=14,ntot=nreal+mch)

C Declare variables

C -----------------

character\*1 kbuf(nbyte1)

character\*4 indat(nbyte4),jbuf(nbyte4)

character\*8 compress,process\_Tb,process\_Ta

character\*40 mapfile(nset)

character\*80 rawamsu,coefile

integer stdout

integer(8) itime

integer idt(nid),ndt(nnd)

integer ichan(mch),lndsea(mpos),ikeepb(mpos),ikeepa(mpos)

integer imx(nset),jmx(nset),ibadc(mch)

real(real\_64) p1,p2,term1,term2,term3,ta(mch,mpos),b,c,clight,ta0

real(real\_64) scale,scale5,scale6,scale9,scale13,scale19

real(real\_64) sctime,counts,rads,rad0,sathgt,rnorm,tb0

real(real\_64) tshelf11a,tshelf12,tshelf2

real(real\_64) soza(mpos),saza(mpos),rlocaz(mpos)

real(real\_64) slat(mpos),slon(mpos)

real(real\_64) cwave(mch),cnst1(mch),cnst2(mch)

real(real\_64) cfrq0(mch)

real(real\_64) rad(mch,mpos),tb(mch,mpos),sfchgt(mpos)

real(real\_64) c0(mch),c1(mch),c2(mch)

real(real\_32) bdata(ntot),adata(ntot)

real(real\_64) f0(mpos,mch),f1(mpos,mch),f2(mpos,mch)

real(real\_64) eta(mch),tref(mch),badr(mch),badtb(mch),dt(mch)

real(real\_64) badta(mch)

C Declare equivalences

C --------------------

equivalence (kbuf(1),jbuf(1))

common/switches/compress,process\_Tb,process\_Ta

C Set information for different resolution map datasets

C (NOTE: Currently we do not use this information)

C -----------------------------------------------------

data imx / 360, 720, 1440 /

data jmx / 181, 361, 721 /

data mapfile / 'mapdat\_100.iee', 'mapdat\_050.iee',

x 'mapdat\_025.iee' /

C Lower/upper limits for gross temperature check on Tb

C ----------------------------------------------------

data tlo,thi / 100., 400. /

C Constants for Planck equation

C -----------------------------

data p1,p2 / 1.1910659d-5, 1.438833d0 /

C Speed of light (m/s) and center frequency of AMSU-A channels (GHz)

C ------------------------------------------------------------------

data clight / 2.99793d8 /

data cfrq0 / 23.8d0, 31.4d0, 50.3d0, 52.8d0, 53.596d0, 54.50d0,

x 54.94d0, 55.50d0, 6\*57.290344d0, 89.0d0 /

C Set constants for Ta to Tb conversion

C -------------------------------------

data eta /.01d0,.08d0,.03d0,.04d0,.04d0,.03d0,.03d0,7\*.04d0,.11d0/

C Missing data flag

C -----------------

data rmiss / -999. /

C Channel numbers

C ---------------

data ichan / 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,

x 12, 13, 14, 15 /

C Set I/O unit numbers (including standard output)

C ------------------------------------------------

data stdout / 6/

data lundx /12/

data lubfrb /52/, lubfra /53/

write(stdout,\*)' '

write(stdout,\*)' BEGIN AMSU-A 1B DECODE'

C Initialize arrays

C -----------------

badr = 0.

badtb = 0.

badta = 0.

nprint = 500 ! skip between data record diagnostic prints

C Write header record to output IEEE file

C ---------------------------------------

write(stdout,\*)' '

write(stdout,\*)'header information below'

write(stdout,\*)'nreal,mch = ',nreal,mch

write(stdout,\*)'ntot = ',ntot

write(stdout,\*)'channel numbers below'

write(stdout,\*) (ichan(i),i=1,mch)

write(stdout,\*)' '

write(lunout) nreal,mch,(ichan(i),i=1,mch)

C Open output BUFR files

C ----------------------

call openbf(lubfrb,'OUT',lundx)

call openbf(lubfra,'OUT',lundx)

C Load arrays containing coefficients to convert the antenna

C temperature (Ta) to brightness temperature (Tb) - finish loading

C after read, then scale coefficients

C -----------------------------------------------------------------

open(luncof,file=coefile)

C Default values for when file is missing

C ---------------------------------------

do n = 1,mch

do i = 1,mpos

f0(i,n)=100.0

f1(i,n)=0.0

f2(i,n)=0.0

end do

end do

do i = 1,mpos

read(luncof,\*,err=1800,end=100) (f0(i,n),f1(i,n),f2(i,n),n=1,5)

end do

do i = 1,mpos

read(luncof,\*,err=1800,end=100)(f0(i,n),f1(i,n),f2(i,n),n=6,10)

end do

write(stdout,\*)'Read Ta to Tb conversion coefficients from ',

& 'coefficient file'

C If present, read eta factor from coefficient file

C -------------------------------------------------

read(luncof,\*,end=101)

write(stdout,\*)'Read eta factor from coefficient file'

read(luncof,\*) (eta(n),n=1,9),eta(15)

do j=10,14

eta(j)=eta(9)

end do

go to 102

100 continue

write(stdout,\*)'Ta to Tb conversion coefficient file missing, ',

& 'use default values --> no Ta to Tb conversion'

101 continue

write(stdout,\*)'Eta factor not found in coefficient file, use ',

& 'default values --> no Ta to Tb conversion'

102 continue

close(luncof)

do i = 1,mpos

f0(i,15) = f0(i,10)

f1(i,15) = f1(i,10)

f2(i,15) = f2(i,10)

end do

do i = 1,mpos

do j = 10,14

f0(i,j) = f0(i,9)

f1(i,j) = f1(i,9)

f2(i,j) = f2(i,9)

end do

end do

do i = 1,mpos

do j = 1,mch

f0(i,j) = f0(i,j)/100.

f1(i,j) = f1(i,j)/100.

f2(i,j) = f2(i,j)/100.

end do

end do

C Open unit to raw 1B AMSU-A data file - read header record, see if

C valid data type - if not, exit routine

C -----------------------------------------------------------------

open(lunin,file=rawamsu,recl=nbyte1/rfac,

& access='direct',status='old')

nri = 1

read (lunin,rec=nri,err=1900) (kbuf(i),i=1,nbyte1)

C Load header record into work array

C ----------------------------------

do i = 1,nbyte4

indat(i) = jbuf(i)

end do

C Extract NOAA spacecraft identification code (72\*8+1=577)

C --------------------------------------------------------

jsat = lbyte(577,16,indat)

if (jsat.eq.4) then ! NOAA-15

jsat0 = jsat

jsat = 15

write(stdout,\*) '\*\*\*WARNING: reset satellite id from ',

x jsat0,' to ',jsat

elseif (jsat.eq.2) then ! NOAA-16

jsat0 = jsat

jsat = 16

write(stdout,\*) '\*\*\*WARNING: reset satellite id from ',

x jsat0,' to ',jsat

elseif (jsat.eq.6) then ! NOAA-17

jsat0 = jsat

jsat = 17

write(stdout,\*) '\*\*\*WARNING: reset satellite id from ',

x jsat0,' to ',jsat

elseif (jsat.eq.7) then ! NOAA-18

jsat0 = jsat

jsat = 18

write(stdout,\*) '\*\*\*WARNING: reset satellite id from ',

x jsat0,' to ',jsat

endif

C Extract data type code (76\*8+1=609)

C -----------------------------------

jtype = lbyte(609,16,indat)

if (jtype.ne.10) then

write(stdout,\*)'\*\*\*ERROR\*\*\* Input data file does not contain',

x ' AMSU-A data (type=10). data type = ',jtype

call w3tage('BUFR\_TRANAMSUA')

call errexit(1)

endif

write(stdout,\*) 'Data and satellite type = ',jtype,jsat

C Extract number of data records in data set (144\*8+1=1153)

C and number of scans (146\*8+1=1169)

C ---------------------------------------------------------

nrecs = lbyte(1153,16,indat)

nscan = lbyte(1169,16,indat)

write(stdout,\*)'nrecs,nscan=',nrecs,nscan

C Extract coefficients for radiance to temperature conversion

C -----------------------------------------------------------

scale6 = 1.d-6

do j = 1,mch

jb = 689 + (j-1)\*12

jb0 = jb

jb1 = jb0 + 4

jb2 = jb1 + 4

cwave(j) = xfloat(1,kbuf(jb0))\*scale6

cnst1(j) = xfloat(1,kbuf(jb1))\*scale6

cnst2(j) = xfloat(1,kbuf(jb2))\*scale6

end do

write(stdout,\*)'chn 1 cwave,cnst1,cnst2=',

x cwave(1),cnst1(1),cnst2(1)

C Extract instrument temperature and load into reference array

C ------------------------------------------------------------

scale = 1.d-2

tshelf11a = mbyte(1745,16,indat)\*scale ! (218\*8+1=1745)

tshelf12 = mbyte(1793,16,indat)\*scale ! (224\*8+1=1793)

tshelf2 = mbyte(1841,16,indat)\*scale ! (230\*8+1=1841)

C A2 antenna handles channels 1,2

C -------------------------------

tref(1) = tshelf2

tref(2) = tshelf2

C A1-1 antenna handles channels 6,7,9-15

C --------------------------------------

tref(6) = tshelf11a

tref(7) = tshelf11a

do i = 9,15

tref(i) = tshelf11a

end do

C A1-2 antenna handles channels 3,4,5,8

C -------------------------------------

tref(3) = tshelf12

tref(4) = tshelf12

tref(5) = tshelf12

tref(8) = tshelf12

C QC instrument temperatures - the lower and upper bounds are

C aribtrary - if unreasonable, replace with 290K (the instrument

C temperature s/b is near 290K)

C ---------------------------------------------------------------

do i = 1,mch

if (tref(i).lt.270 .or. tref(i).gt.310) tref(i) = 290.

end do

C Extract cold space fixed bias correction

C ----------------------------------------

scale = 1.d-3

do i = 1,mch

jb = (270 + (i-1)\*8)\*8 + 1

dtcold = mbyte(jb,16,indat)\*scale

dt(i) = dtcold

end do

write(stdout,\*)'instrument T=',(tref(i),i=1,mch)

write(stdout,\*)'cold space dt=',(dt(i),i=1,mch)

write(stdout,\*)' '

C Prepatory initializations prior to reading in satellite data

C ------------------------------------------------------------

nopos = 0

nqcbad = 0

nqctim = 0

nqccal = 0

nqcloc = 0

nermin = 0

nermaj = 0

nbadc = 0

nbadl = 0

nbadtb = 0

nbadta = 0

nbadr = 0

nrecb = 0

nreca = 0

nrepb = 0

nrepa = 0

nskipc = 0

nskipq = 0

nskipm = 0

nskiptb = 0

nskipta = 0

nlandb = 0

nlanda = 0

nseab = 0

nseaa = 0

nlo = 0

1200 continue

C\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C MAIN LOOP OVER NUMBER OF SCANS

C\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

nri = nri + 1 ! Increment record counter

C Read data record and load into local work array

C -----------------------------------------------

read(lunin,rec=nri,err=1600) (kbuf(i),i=1,nbyte1)

do i = 1,nbyte4

indat(i) = jbuf(i)

end do

nlo = nlo + 1 ! Increment line counter

line = nlo

C Extract scan line number, date/time, position, and type

C -------------------------------------------------------

iline = lbyte(1,16,indat)

iyear = lbyte(17,16,indat) ! (2\*8+1=17)

iddd = lbyte(33,16,indat) ! (4\*8+1=33)

itime = lbyte(65,32,indat) ! (8\*8+1=65)

sctime = 1.d-3\*itime

jyd = 1000\*iyear + iddd

idt(1) = sctime + 0.5

idt(2) = mod(jyd,1000)

idt(3) = jyd/1000

call dattim(idt,ndt)

C Extract quality bits - if all good (=0) continue, else skip this scan

C ---------------------------------------------------------------------

isum = 0

iqcbad = lbyte(193,1,indat) ! (8\*24+1=193)

iqctim = lbyte(194,1,indat) ! (8\*24+1+1=194)

iqccal = lbyte(196,1,indat) ! (8\*24+1+3=196)

iqcloc = lbyte(197,1,indat) ! (8\*24+1+4=197)

iermin = lbyte(222,1,indat) ! (8\*24+1+29=222)

iermaj = lbyte(223,1,indat) ! (8\*24+1+30=223)

isum = iqcbad + iqctim + iqccal + iqcloc

x + iermin + iermaj

if (iqcbad.ne.0) nqcbad = nqcbad + 1

if (iqctim.ne.0) nqctim = nqctim + 1

if (iqccal.ne.0) nqccal = nqccal + 1

if (iqcloc.ne.0) nqcloc = nqcloc + 1

if (iermin.ne.0) nermin = nermin + 1

if (iermaj.ne.0) nermaj = nermaj + 1

if (isum.ne.0) then

nskipq = nskipq + 1

write(stdout,1000) nlo,iline,iqcbad,iqctim,

x iqccal,iqcloc,iermin,iermaj,nskipq

1000 format('\*\*\*FAIL QC : nlo=',i6,' iline=',i6,

x ' bad=',i2,' time=',i2,' cali=',i2,' loc=',i2,

x ' min=',i2,' maj=',i2,' nskipq=',i6)

goto 1200

endif

C Extract calibration quality bits for each channel

C -------------------------------------------------

isum = 0

do j = 1,mch

ibadc(j) = 0

jb = (32 + (j-1)\*2)\*8+1

iqccal = lbyte(jb,16,indat)

ibadc(j) = iqccal

isum = isum + ibadc(j)

end do

if (isum.ne.0) then

nskipc = nskipc + 1

write(stdout,1005) nlo,iline,(ibadc(j),j=1,mch)

1005 format('\*\*\*FAIL CAL : nlo=',i6,' iline=',i6,

x ' badcal=',20(i2,1x))

endif

C Extract calibration coefficients

C --------------------------------

scale19 = 1.d-19

scale13 = 1.d-13

scale9 = 1.d-9

do j = 1,mch

jb2 = 81 + (j-1)\*12

c2(j) = xfloat(1,kbuf(jb2))\*scale19

jb1 = jb2 + 4

c1(j) = xfloat(1,kbuf(jb1))\*scale13

jb0 = jb1 + 4

c0(j) = xfloat(1,kbuf(jb0))\*scale9

C Code to pull out secondary calibration coefficients

C ---------------------------------------------------

ccccc jb2 = 261 + (j-1)\*12

ccccc c2j = xfloat(1,kbuf(jb2))\*scale19

ccccc jb1 = jb2 + 4

ccccc c1j = xfloat(1,kbuf(jb1))\*scale13

ccccc jb0 = jb1 + 4

ccccc c0j = xfloat(1,kbuf(jb0))\*scale9

end do

C EXTRACT NAVIGATION DATA

C -----------------------

C -----------------------

C Extract spacecraft altitude (km)

C --------------------------------

scale = 1.d-1

jb = 470\*8+1

sathgt = lbyte(jb,16,indat)\*scale

C Extract angular relationships

C -----------------------------

scale = 1.d-2

do i = 1,mpos

jb0 = 472\*8+1 + (i-1)\*48

jb1 = jb0 + 16

jb2 = jb1 + 16

soza(i) = mbyte(jb0,16,indat)\*scale

saza(i) = mbyte(jb1,16,indat)\*scale

rlocaz(i) = mbyte(jb2,16,indat)\*scale

end do

C Extract earth location data

C ---------------------------

scale = 1.d-4

do i = 1,mpos

jb0 = 653 + (i-1)\*8

slat(i) = xfloat(1,kbuf(jb0))\*scale

jb1 = jb0 + 4

slon(i) = xfloat(1,kbuf(jb1))\*scale

lndsea(i) = rmiss

sfchgt(i) = rmiss

ikeepb(i) = 0

ikeepa(i) = 0

if ( (abs(slat(i)).gt.90.) .or.

x (abs(slon(i)).gt.180.) ) then

ikeepb(i) = 0

ikeepa(i) = 0

nbadl = nbadl + 1

elseif ( (abs(slat(i)).le.eps) .and.

x (abs(slon(i)).le.eps) ) then

ikeepb(i) = 0

ikeepa(i) = 0

nopos = nopos + 1

else

ikeepb(i) = 1

ikeepa(i) = 1

C Set surface type information based on resolution option

C If iresol = 1, use 1.0 degree dataset

C iresol = 2, use 0.5 degree dataset

C iresol = 3, use 0.25 degree dataset

C -------------------------------------------------------

ccccc call lansea3(xlat,xlon,imx(iresol),jmx(iresol),

ccccc x mapfile(iresol),rmask,water,elev,stdev)

ccccc lndsea(i) = rmask + 1.d-3

ccccc sfchgt(i) = elev

ils = lansea(slat(i),slon(i),ll)

if (ils.eq.2) then

lndsea(i) = 0

sfchgt(i) = 0.0

elseif (ils.eq.1) then

lndsea(i) = 1

sfchgt(i) = 1.\*ll

else

lndsea(i) = rmiss

sfchgt(i) = rmiss

endif

endif

end do

C Is scan line for full scan mode - if so, keep, else skip this scan

C ------------------------------------------------------------------

imode1 = lbyte(7207,1,indat) ! (8\*900+1+6=7207)

imode2 = lbyte(17511,1,indat) ! (8\*2188+1+6=17511)

if ( imode1.ne.1 .or. imode2.ne.1) then

nskipm = nskipm + 1

write(stdout,1010) nlo,iline,imode1,imode2,nskipm

1010 format('\*\*\*FAIL MODE: nlo=',i6,' iline=',i6,

x ' imode1=',i2,' imode2=',i2,' nskipm=',i6)

goto 1200

endif

C Extract AMSU-A counts for channels 3-15 and 1-2, then convert counts

C to radiances

C --------------------------------------------------------------------

do i = 1,mpos

C Channels 3-15

C -------------

jb = (904 + (i-1)\*34 + 4\*2)\*8+1

do j = 3,mch

jb0 = jb + (j-3)\*16

counts = lbyte(jb0,16,indat)

rads = c0(j) + (c1(j)+c2(j)\*counts)\*counts

if (rads.lt.0.) then

nbadr = nbadr + 1

badr(j) = badr(j) + 1

rads = rmiss

endif

rad(j,i) = rads

end do

C Channels 1-2

C ------------

jb = (2192 + (i-1)\*8 + 2\*2)\*8+1

do j = 1,2

jb0 = jb + (j-1)\*16

counts = lbyte(jb0,16,indat)

rads = c0(j) + (c1(j)+c2(j)\*counts)\*counts

if (rads.lt.0.) then

nbadr = nbadr + 1

badr(j) = badr(j) + 1

rads = rmiss

endif

rad(j,i) = rads

end do

end do

C-----------------------------------------------------------------------

C Convert radiances to antenna temperature (Ta), then convert antenna

C temperature (Ta) to brightness temperature (Tb)

C QC all channels - if all channels are bad for a given spot, set

C flag to omit data in final write (for both Tb and Ta)

C-----------------------------------------------------------------------

do i = 1,mpos

ibadtb = 0

ibadta = 0

do j = 1,mch

rads = rad(j,i)

term1 = p2\*cwave(j)

if (rads.gt.eps) then

term2 = 1. + p1\*cwave(j)\*\*3/rads

if (term2.le.0) term2 = eps

term3 = log(term2)

ta0 = term1/term3

b = cnst1(j)

c = cnst2(j)

ta(j,i) = (ta0-b)/c

rnorm = eta(j)\*f1(i,j) + f2(i,j) + f0(i,j)

tb(j,i) = (rnorm\*ta(j,i) - eta(j)\*f1(i,j)\*tref(j) -

x f2(i,j)\*(2.73+dt(j)))/f0(i,j)

tb0 = tb(j,i)

else

ta(j,i) = rmiss

tb(j,i) = rmiss

tb0 = rmiss

endif

C Apply gross check to Tb using limits TLO and THI set in data stmt

C -----------------------------------------------------------------

if ( (tb(j,i).lt.tlo) .or.

x (tb(j,i).gt.thi) ) then

nbadtb = nbadtb + 1

badtb(j) = badtb(j) + 1

tb(j,i) = rmiss

endif

C Apply gross check to Ta - only limit is for Ta > 1000

C -----------------------------------------------------

if (ta(j,i).gt.1000.) then

nbadta = nbadta + 1

badta(j) = badta(j) + 1

ta(j,i) = rmiss

endif

C If calibration quality flag for this channel is nonzero, we do not

C want to use this channel for Tb

C ------------------------------------------------------------------

if (ibadc(j).ne.0) then

nbadc = nbadc + nbadc

tb(j,i) = rmiss

endif

C Count number of bad channels for current scan position

C ------------------------------------------------------

if (tb(j,i).lt.0.) ibadtb = ibadtb + 1

if (ta(j,i).lt.0.) ibadta = ibadta + 1

end do

C If all Tb channels are bad for current scan position, set keep flag

C to zero (this tells the code below to not write Tb for this spot

C to the output files)

C -------------------------------------------------------------------

if (ibadtb.eq.mch) then

nskiptb = nskiptb + 1

ikeepb(i) = 0

endif

C If all Ta channels are bad for current scan position, set keep flag

C to zero (this tells the code below to not write Ta for this spot

C to the output BUFR file)

C -------------------------------------------------------------------

if (ibadta.eq.mch) then

nskipta = nskipta + 1

ikeepa(i) = 0

endif

end do

C Write AMSU-A data for each spot position on current scan line

C -------------------------------------------------------------

do i = 1,mpos

if (min(ikeepb(i),ikeepa(i)).eq.1) then

bdata(1) = jsat

bdata(2) = jtype

bdata(3) = ndt(1)

bdata(4) = ndt(2)

bdata(5) = ndt(3)

bdata(6) = 3600\*ndt(4) + 60\*ndt(5) + ndt(6)

bdata(7) = lndsea(i)

bdata(8) = i

bdata(9) = slat(i)

bdata(10)= slon(i)

ccccc bdata(11)= rlocaz(i)

bdata(11)= saza(i)

bdata(12)= soza(i)

bdata(13)= sfchgt(i)

bdata(14)= sathgt

adata(1:14) = bdata(1:14)

if(process\_Tb.eq.'YES') then

if (ikeepb(i).eq.1) then

if (lndsea(i).lt.0.5) nseab = nseab + 1

if (lndsea(i).gt.0.5) nlandb = nlandb + 1

do j = 1,mch

bdata(14+j) = tb(j,i)

end do

nrecb = nrecb + 1

write(lunout) (bdata(j),j=1,ntot) ! ieee write

call bufr1b(lubfrb,'NC021023',nreal,mch,bdata,nrepb)

endif

endif

if(process\_Ta.eq.'YES') then

if (ikeepa(i).eq.1) then

if (lndsea(i).lt.0.5) nseaa = nseaa + 1

if (lndsea(i).gt.0.5) nlanda = nlanda + 1

do j = 1,mch

adata(14+j) = ta(j,i)

end do

nreca = nreca + 1

call bufr1b(lubfra,'NC021123',nreal,mch,adata,nrepa)

endif

endif

endif

end do

C Every NPRINT scan lines, print mpos-th record

C ---------------------------------------------

if (mod(nlo,nprint).eq.0) then

if(process\_Tb.eq.'YES') then

write(stdout,\*)' '

write(stdout,\*)' Tb data for line,rec=',nlo,nrecb

write(stdout,\*) (bdata(i),i=1,ntot)

write(stdout,\*)' '

endif

if(process\_Ta.eq.'YES') then

write(stdout,\*)' '

write(stdout,\*)' Ta data for line,rec=',nlo,nreca

write(stdout,\*) (adata(i),i=1,ntot)

write(stdout,\*)' '

endif

endif

C\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C DONE WITH THIS SCAN LINE, READ NEXT SCAN LINE

C\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

goto 1200

1600 continue

C All scan lines have been read and processed, summarize

C ------------------------------------------------------

write(stdout,\*)' '

write(stdout,\*)'Done reading raw 1b file'

write(stdout,\*)' '

write(stdout,\*)'AMSU-A INGEST STATS:'

write(stdout,\*)' no. scan lines = ',nlo,nrecs,nscan

write(stdout,\*)' no. fail good qc = ',nqcbad

write(stdout,\*)' no. fail time qc = ',nqctim

write(stdout,\*)' no. fail cali qc = ',nqccal

write(stdout,\*)' no. fail loc qc = ',nqcloc

write(stdout,\*)' no. frame errors = ',nermin,nermaj

write(stdout,\*)' no. scans bad qc = ',nskipq

write(stdout,\*)' no. scans bad calibr. = ',nskipc

write(stdout,\*)' no. scans bad mode = ',nskipm

write(stdout,\*)' no. bad (lat,lon) = ',nbadl

write(stdout,\*)' no. zero (lat,lon) = ',nopos

write(stdout,\*)' no. bad radiances = ',nbadr

if(process\_Tb.eq.'YES') then

write(stdout,\*)' no. bad calibration = ',nbadc

write(stdout,\*)' no. scans with bad Tb = ',nskiptb

write(stdout,\*)' no. bad Tb values = ',nbadtb

write(stdout,\*)' no. land/sea Tb obs = ',nlandb,nseab

write(stdout,\*)' no. Tb recs written = ',nrecb

write(stdout,\*)' no. Tb BUFR rpts written = ',nrepb

endif

if(process\_Ta.eq.'YES') then

write(stdout,\*)' no. scans with bad Ta = ',nskipta

write(stdout,\*)' no. bad Ta values = ',nbadta

write(stdout,\*)' no. land/sea Ta obs = ',nlanda,nseaa

write(stdout,\*)' no. Ta recs written = ',nreca

write(stdout,\*)' no. Ta BUFR rpts written = ',nrepa

endif

write(stdout,\*)' '

write(stdout,\*)'bad radiance,temperature counts per channel'

write(stdout,1020)

1020 format(t1,'channel',t10,'bad rad',t20,'bad Tb',t30,'bad Ta')

sumr = 0.

sumtb = 0.

sumta = 0.

do j = 1,mch

write(stdout,1030) j,badr(j),badtb(j),badta(j)

1030 format(t1,i2,t10,f8.1,t20,f8.1,t30,f8.1)

sumr = sumr + badr(j)

sumtb = sumtb + badtb(j)

sumta = sumta + badta(j)

end do

write(stdout,\*)'nbadr,nbadtb,nbadta=',sumr,sumtb,sumta

write(stdout,\*)' '

write(stdout,\*)' AMSU-A 1B DECODE COMPLETED'

write(stdout,\*)' '

C Close UNITs

C -----------

close(lunin)

close(lunout)

call closbf(lubfrb)

call closbf(lubfra)

call system('echo YES > Tb')

if(process\_Tb.eq.'YES') then

if(nrecb.eq.0) then

write(stdout,1003)

1003 format(/' NO Tb RECORDS WRITTEN -- DISABLING ALL ',

1 'SUBSEQUENT Tb PROCESSING.'/)

call system('echo NO > Tb')

else

call mesgbc(lubfrb,msgt,icomp)

if(icomp.eq.1) then

print'(/"OUTPUT Tb BUFR FILE MESSAGES '//

. 'C O M P R E S S E D"/"FIRST MESSAGE TYPE FOUND IS",I5/)',

. msgt

elseif(icomp.eq.0) then

print'(/"OUTPUT Tb BUFR FILE MESSAGES '//

. 'U N C O M P R E S S E D"/"FIRST MESSAGE TYPE FOUND IS",'//

. 'I5/)', msgt

elseif(icomp.eq.-1) then

print'(//"ERROR READING OUTPUT Tb BUFR FILE - MESSAGE '//

. 'COMPRESSION UNKNOWN"/)'

elseif(icomp.eq.-3) then

print'(/"OUTPUT Tb BUFR FILE DOES NOT EXIST"/)'

elseif(icomp.eq.-2) then

print'(/"OUTPUT Tb BUFR FILE HAS NO DATA MESSAGES"/'//

. '"FIRST MESSAGE TYPE FOUND IS",I5/)', msgt

endif

endif

endif

call system('echo YES > Ta')

if(process\_Ta.eq.'YES') then

if(nreca.eq.0) then

write(stdout,1004)

1004 format(/' NO Ta RECORDS WRITTEN -- DISABLING ALL ',

. 'SUBSEQUENT Ta PROCESSING.'/)

call system('echo NO > Ta')

else

call mesgbc(lubfra,msgt,icomp)

if(icomp.eq.1) then

print'(/"OUTPUT Ta BUFR FILE MESSAGES '//

. 'C O M P R E S S E D"/"FIRST MESSAGE TYPE FOUND IS",I5/)',

. msgt

elseif(icomp.eq.0) then

print'(/"OUTPUT Ta BUFR FILE MESSAGES '//

. 'U N C O M P R E S S E D"/"FIRST MESSAGE TYPE FOUND IS",'//

. 'I5/)', msgt

elseif(icomp.eq.-1) then

print'(//"ERROR READING OUTPUT Ta BUFR FILE - MESSAGE '//

. 'COMPRESSION UNKNOWN"/)'

elseif(icomp.eq.-3) then

print'(/"OUTPUT Ta BUFR FILE DOES NOT EXIST"/)'

elseif(icomp.eq.-2) then

print'(/"OUTPUT Ta BUFR FILE HAS NO DATA MESSAGES"/'//

. '"FIRST MESSAGE TYPE FOUND IS",I5/)', msgt

endif

endif

endif

close(lubfrb)

close(lubfra)

return

C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

C ERRORS

C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

C Error reading Ta to Tb coefficient file

C ---------------------------------------

1800 continue

write(stdout,\*)'\*\*\* error reading coefficient file ',coefile

close(luncof)

close(lunin)

close(lunout)

call closbf(lubfrb)

call closbf(lubfra)

call w3tage('BUFR\_TRANAMSUA')

call errexit(2)

C Error reading 1B file

C ---------------------

1900 continue

write(stdout,\*)' \*\*\* error reading hdr record of file ',rawamsu

close(luncof)

close(lunin)

close(lunout)

call closbf(lubfrb)

call closbf(lubfra)

call w3tage('BUFR\_TRANAMSUA')

call errexit(3)

end

SUBROUTINE CHARS(IWORD,LEN,CWORD)

C$$$ SUBPROGRAM DOCUMENTATION BLOCK

C

C SUBPROGRAM: CHARS

C PRGMMR: BERT KATZ ORG: NP20 DATE: 1997-11-06

C

C ABSTRACT: Turns integer into character string of specified length,

C starting at low-order byte of integer.

C

C PROGRAM HISTORY LOG:

C 1997-11-06 Katz -- Original author

C

C USAGE: CALL CHARS(IWORD,LEN,CWORD)

C INPUT ARGUMENT LIST:

C IWORD - INTEGER argument

C LEN - INTEGER argument holding number of low-order bytes

C of first argument to convert into character

C

C OUTPUT ARGUMENT LIST:

C CWORD - CHARACTER argument

C

C REMARKS:

C

C ATTRIBUTES:

C LANGUAGE: FORTRAN 90

C MACHINE: NCEP-CCS

C

C$$$

character\*1 cword(len)

integer iword

do ic = len , 1 , -1

ibeg = (len - ic) \* 8

ichr = ibits(iword,ibeg,8)

cword(ic) = char(ichr)

enddo

return

end

SUBROUTINE DATTIM(IDT,NDT)

C$$$ SUBPROGRAM DOCUMENTATION BLOCK

C

C SUBPROGRAM: DATTIM

C PRGMMR: BERT KATZ ORG: NP20 DATE: 1997-11-06

C

C ABSTRACT: Converts year, day-of-year, and second-of-day into year,

C month-of-year, day-of-month, hour-of-day, minute-of-hour, and

C second-of-minute.

C

C PROGRAM HISTORY LOG:

C 1997-11-06 Katz -- Original author

C

C USAGE: CALL DATTIM(IDT,NDT)

C INPUT ARGUMENT LIST:

C IDT - INTEGER array argument containing three members:

C year, day-of-year, and second-of-day.

C

C OUTPUT ARGUMENT LIST:

C KDT - INTEGER array argument containing six members:

C year, month-of-year, day-of-month, hour-of-day,

C minute-of-hour, and second-of-minute.

C

C REMARKS:

C NONE

C

C ATTRIBUTES:

C LANGUAGE: FORTRAN 90

C MACHINE: NCEP-CCS

C

C$$$

integer idt(3),ndt(6)

integer iday,ihr,imin,imon,isec,iyr,jday,jsec

external w3fs26

intrinsic mod

JULIAN(IYR,IDYR) = -31739 + 1461 \* (IYR + 4799) / 4

& -3 \* ((IYR + 4899) / 100) / 4 + IDYR

iyr = idt(3)

jday = idt(2)

C If year is two digits, convert to 4 digits

C ------------------------------------------

if (iyr.ge.0.and.iyr.le.99) then

if (iyr.lt.21) then

kyr = iyr + 2000

else

kyr = iyr + 1900

endif

else

kyr = iyr

endif

C Compute julian day number as number days after 4713 bc

C ------------------------------------------------------

idy = jday

jdn = julian(kyr,idy)

call w3fs26(jdn,iyear,jmo,jda,idaywk,idayyr)

imon = jmo

iday = jda

jsec = idt(1)

ihr = jsec/3600

imin = mod(jsec,3600)/60

isec = jsec - 3600\*ihr - 60\*imin

ndt(1) = iyr

ndt(2) = imon

ndt(3) = iday

ndt(4) = ihr

ndt(5) = imin

ndt(6) = isec

return

end

INTEGER FUNCTION ICHARS(CWORD,LEN)

C$$$ SUBPROGRAM DOCUMENTATION BLOCK

C

C SUBPROGRAM: ICHARS

C PRGMMR: BERT KATZ ORG: NP20 DATE: 1997-11-05

C

C ABSTRACT: Turns character string of specified length into integer.

C

C PROGRAM HISTORY LOG:

C 1997-11-05 Katz -- Original author

C

C USAGE: ICHARS(CWORD,LEN)

C INPUT ARGUMENT LIST:

C CWORD - CHARACTER\*1 array argument

C LEN - INTEGER argument holding length of cword

C

C REMARKS:

C NONE

C

C ATTRIBUTES:

C LANGUAGE: FORTRAN 90

C MACHINE: NCEP-CCS

C

C$$$

character\*1 cword(len)

lchars = 0

do ic = len , 1 , -1

ibeg = (len - ic) \* 8

icmove = mova2i(cword(ic))

call mvbits(icmove,0,8,lchars,ibeg)

enddo

ichars = lchars

return

end

cfpp$ expand(ichars,lbit)

INTEGER FUNCTION LANSEA(RLAT,RLON,LEVEL)

C$$$ SUBPROGRAM DOCUMENTATION BLOCK

C

C SUBPROGRAM: LANSEA

C PRGMMR: BERT KATZ ORG: NP20 DATE: 1997-11-05

C

C ABSTRACT: Calculates topography, land/sea status from latitude and

C longitude.

C

C PROGRAM HISTORY LOG:

C 1997-11-05 Katz -- Original author

C

C USAGE: LANSEA(RLAT,RLON,LEVEL)

C INPUT ARGUMENT LIST:

C RLAT - INTEGER argument containing scaled latitude

C RLON - INTEGER argument containing scaled longitude

C

C OUTPUT ARGUMENT LIST:

C LEVEL - INTEGER argument containing scaled topography

C

C INPUT FILES:

C UNIT 41 - Binary low-resolution topography file

C

C REMARKS:

C NONE

C

C ATTRIBUTES:

C LANGUAGE: FORTRAN 90

C MACHINE: NCEP-CCS

C

C$$$

include 'rfac.inc'

integer,parameter::real\_64=selected\_real\_kind(15,307)

integer ilat,ilon,level

real(real\_64) rlat,rlon

real slat,slon,xlon

integer iopn,iu,lan,last,lat,lenr,lon

character\*12 name

character\*4 iflag(12),kelev(192)

character\*2 ielev(360)

integer lbit,ichars

external lbit,ichars

intrinsic float,max0

equivalence (iflag(1),kelev(1)), (ielev(1),kelev(13))

data name/'lowtopog.dat'/,iu/41/,lenr/768/,last/0/,iopn/0/

save iopn,kelev,last

if (iopn.eq.0) then

open (iu,recl=lenr/rfac,

& file=name,access='direct',status='old')

iopn = 1

endif

lan = 0

level = 0

slat = rlat

slon = rlon

lat = slat + 1.

if (slat.lt.0.) lat = slat

lat = max0(lat,-87)

lat = 91 - lat

if (lat.eq.last) go to 100

read (iu,rec=lat) kelev

last = lat

100 xlon = slon

if (xlon.lt.0.) xlon = xlon + 360.

lon = xlon

if (lon.eq.360) lon = 0

lon = lon + 1

lan = lbit(lon,iflag)

if (lan.ne.0) then

ltemp = ichars(ielev(lon),2)

if (btest(ltemp,15)) then

ltemp = ior(ltemp,-65536)

endif

level = ltemp

endif

lansea = 2 - lan

return

end

cfpp$ expand(ichars)

INTEGER FUNCTION LBIT(J,ARRAY)

C$$$ SUBPROGRAM DOCUMENTATION BLOCK

C

C SUBPROGRAM: LBIT

C PRGMMR: BERT KATZ ORG: NP20 DATE: 1997-11-05

C

C ABSTRACT: Extracts j'th bit from array of CHARACTER\*4.

C

C PROGRAM HISTORY LOG:

C 1997-11-05 Katz -- Original author

C

C USAGE: LBIT(J,ARRAY)

C INPUT ARGUMENT LIST:

C J - INTEGER argument

C ARRAY - CHARACTER\*4 array argument

C

C REMARKS:

C NONE

C

C ATTRIBUTES:

C LANGUAGE: FORTRAN 90

C MACHINE: NCEP-CCS

C

C$$$

integer j

character\*4 array(\*)

integer ibit,jout,jw,jword,nbit

integer ichars

external ichars

intrinsic btest

jw = (j-1)/32

nbit = j - jw\*32

jword = ichars(array(jw+1),4)

ibit = 32 - nbit

jout = 0

if (btest(jword,ibit)) jout = 1

lbit = jout

return

end

cfpp$ expand(ichars)

INTEGER FUNCTION MBYTE(J,LENGTH,JARRAY)

C$$$ SUBPROGRAM DOCUMENTATION BLOCK

C

C SUBPROGRAM: MBYTE

C PRGMMR: BERT KATZ ORG: NP20 DATE: 1997-11-05

C

C ABSTRACT: Extracts bit string from array of CHARACTER\*4 and

C converts it to INTEGER. Entry point MBYTE propagates sign bit

C in result; entry point LBYTE does not.

C

C PROGRAM HISTORY LOG:

C 1997-11-05 Katz -- Original author

C

C USAGE: MBYTE(J,LENGTH,JARRAY)

C INPUT ARGUMENT LIST:

C J - INTEGER argument containing starting bit

C LENGTH - integer argument containing number of bits

C (maximum value 32)

C JARRAY - CHARACTER\*4 array argument

C

C OUTPUT FILES:

C UNIT 06 - printout

C

C REMARKS:

C NONE

C

C ATTRIBUTES:

C LANGUAGE: FORTRAN 90

C MACHINE: NCEP-CCS

C

C$$$

integer j,length

character\*4 jarray(\*)

integer inleft,jbit,kompl,mflag,n,nlj,nrj,

+ nsh,nword

integer(8) item,jleft,jrite,mask

integer(8) kounts(33)

integer ichars

external ichars

intrinsic iand,ior,mod

integer lbyte

data kounts/1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,

+ 16384,32768,65536,131072,262144,524288,1048576,2097152,

+ 4194304,8388608,16777216,33554432,67108864,134217728,

+ 268435456,536870912,1073741824,2147483648\_8,0/

C ENTRY MBYTE

C -----------

mflag = 1

110 nword = (j-1)/32 + 1

if (length.lt.1 .or. length.gt.32) write (\*,fmt=120) length

120 format (' improper byte length in mbyte or lbyte',i10)

nlj = mod(j-1,32)

nrj = 32 - length - nlj

if (nrj.lt.0) go to 150

item = ichars(jarray(nword),4)

kompl = 33 - length - nlj

mask = -kounts(kompl)

item = iand(item,mask)

item = item/kounts(nrj+1)

mask = kounts(length+1) - 1

item = iand(item,mask)

130 if (mflag.eq.0) go to 140

c ... means logical byte

mbyte = item

jbit = iand(kounts(length),item)

if (jbit.eq.0) return

c ... need sign extension

mask = -mask - 1

item = ior(item,mask)

mbyte = item

return

C ENTRY LBYTE

C -----------

entry lbyte(j,length,jarray)

mflag = 0

go to 110

140 lbyte = item

return

c ... byte spans two words

150 inleft = length + nrj

mask = kounts(inleft+1) - 1

jleft = ichars(jarray(nword),4)

jleft = iand(jleft,mask)

nsh = 1 - nrj

jleft = jleft\*kounts(nsh)

n = 1 - nrj

jrite = ichars(jarray(nword+1),4)

kompl = 33 + nrj

mask = -kounts(kompl)

jrite = iand(jrite,mask)

jrite = jrite/kounts(nrj+33)

mask = kounts(n) - 1

jrite = iand(jrite,mask)

item = ior(jrite,jleft)

mask = kounts(length+1) - 1

go to 130

end

cfpp$ expand(ichars)

REAL FUNCTION XFLOAT(JB,IARRAY)

C$$$ SUBPROGRAM DOCUMENTATION BLOCK

C

C SUBPROGRAM: XFLOAT

C PRGMMR: BERT KATZ ORG: NP20 DATE: 1997-11-05

C

C ABSTRACT: Takes two consecutive elements of CHARACTER\*2 array and

C forms a floating point number from them.

C

C PROGRAM HISTORY LOG:

C 1997-11-05 Katz -- Original author

C

C USAGE: XFLOAT(JB,IARRAY)

C INPUT ARGUMENT LIST:

C JB - INTEGER argument containing array location

C IARRAY - CHARACTER\*2 array argument

C

C REMARKS:

C NONE

C

C ATTRIBUTES:

C LANGUAGE: FORTRAN 90

C MACHINE: NCEP-CCS

C

C$$$

integer jb

integer(8) mask

data mask/x'ffffffff00000000'/

ccccc character\*2 iarray(2)

character\*2 iarray(2000)

character\*4 conv

real xf

integer j

integer(8) jj

integer in(2)

integer ichars

external ichars

intrinsic btest,ior

j = jb

conv(1:2) = iarray(j)

conv(3:4) = iarray(j+1)

jj = ichars(conv,4)

if (btest(jj,31\_8)) then

jj = ior(jj,mask)

endif

xf = jj

xfloat = xf

return

end