LISTING OF VERSION 1 OF THE GLI CLOUD MASK

COOPERATIVE INSTITUTE FOR METEOROLOGICAL SATELLITE STUDIES
Space Science and Engineering Center
University of Wisconsin—Madison
1225 West Dayton Street
Madison, Wisconsin 53706
Telex: 265 452 UOFWISC MDS

CIMSS

October 20, 1998

Dr. Gary Toller
SAIC/GSC
6100 Chevy Chase Dr.
Suite 200
Laurel, Md. 20707-2929
Tel. 301 953-2700
FAX 301 953-1213
toller@mrslate.gsc.saic.com
adeos2@cpqm.saic.com

Dear Dr. Toller;

Enclosed is our April 1997 interim report for our cloud masking project for GLI. The listing of the version 1 code of the masking algorithm is also enclosed as required. The version 1 code has been delivered to EORC and as passed preliminary testing, we are currently updating this code for a next delivery.

Sincerely

Steven A. Ackerman

LISTING OF VERSION 1 OF THE GLI CLOUD MASK

COOPERATIVE INSTITUTE FOR METEOROLOGICAL SATELLITE STUDIES
Space Science and Engineering Center
University of Wisconsin—Madison
1225 West Dayton Street
Madison, Wisconsin 53706
Telex: 265 452 UOFWISC MDS

CIMSS

October 20, 1998

Dr. Gary Toller
SAIC/GSC
6100 Chevy Chase Dr.
Suite 200
Laurel, Md. 20707-2929
Tel. 301 953-2700
FAX 301 953-1213
toller@mrslate.gsc.saic.com
adeos2@cpqm.saic.com

Dear Dr. Toller;

Enclosed is our April 1997 interim report for our cloud masking project for GLI. The listing of the version 1 code of the masking algorithm is also enclosed as required. The version 1 code has been delivered to EORC and as passed preliminary testing, we are currently updating this code for a next delivery.

Sincerely

Steven A. Ackerman

First 1998 interim report for the Pre-launch Standard Product Cloud Mask Algorithm Development for GLI aboard ADEOS II. This report covers first quarter research activities of the GLI cloud masking algorithm (ATSK1, ATSK2) development.

Description of Activities

During this period we continued to test the algorithm on MAS data test-bed.

Issues and Problems

We continue to test the impact of the 1.64 μm channel on the cloud masking algorithm.

Algorithm Status

Algorithm Theoretical Basis Document has been written which describes the GLI cloud. It now needs to be updated to Version-2.

Activities Planned for the Next Reporting Period:

- Continued validation of the GLI cloud mask through comparison with lidar observations,
- Comparison of GLI mask with MODIS cloud mask using MAS observations,
- Better define the impact of excluding the 1.64 μm channel in the algorithm.
- preparation of the next version of the GLI cloud mask, and the ATBD document

Summary of Percent Completion

The preliminary cloud/clear detection algorithm has been developed and delivered to EORC. Revisions and continued validation studies of the algorithm need to be conducted next period. We are approximately 65% complete with the project.

Steve Ackerman
SSEC/CIMSS/AOS
1225 W. Dayton Street
University of Wisconsin-Madison
Madison, WI 53706
stevea@ssec.wisc.edu
Phone: 608-263-3647
Fax: 608-262-5974

LISTING OF VERSION 1 OF THE GLI CLOUD MASK

```
      program masglimsk

c     A program which creates a cloud mask from MAS spectral data. Output
c     is a series of 16 "bit" flags for each MAS FOV. The flags are arranged
c     into 1432-byte records and written to a binary direct access file, one
c     record for each scan line. Some ancillary data is included in the first
c     record.

c     Input data is stored in HDF-format files.

c     Input ancillary filenames are defined in the "file_open.f" subroutine.

c     This version of the MAS cloud mask processes the data pixel by pixel,
c     but in the context of surrounding observations. The size of the "context",
c     or region, is given by the variables 'nlcntx' (# lines in context) and
c     'necntx' (# pixels in context). The sizes must be odd integer numbers and
c     may not both be set to 1.

c**********************************************************************

c     Command line positional parameters:

c         input HDF file name
c         output file name
c         beginning scan line #
c         total # scan lines to process
c         beginning pixel #
c         total # pixels to process
c         ecosystem file index (1=NA 1km, otherwise global 10 minute)
c         pw value (g/cm-2) for tri-spectral cloud test
c            (optional - input value < 1.0 to disable)
c            ** must enter real number **
c
c**********************************************************************

c      MAS Bands used: visible
c          Mas         Array        Central                GLI
c        Channel#      index #     Wavelength (um)
c           1            1             .56
c           2            2             .66
c           7            3             .88
c           9            4             .945
c          10            5            1.62
c          15            6            1.88
c          20            7            2.14
c
c      MAS Bands used: infrared
c          Mas         Array        Central
c        Channel#      index #     Wavelength (um)
c          31            8            3.75
c          32            9            3.90
c          42           10            8.54
c          45           11           10.98
c          46           12           11.93
c          49           13           13.66

c   *** CAUTION: channel numbers may change. Use array 'inchan' to vary
c                channel numbers corresponding to particular MAS spectral
c                bands. See data statement below.

c**********************************************************************

c     Set constants.

c     Set # lines and # elements for regional "context". Do not set
c     both 'nlcntx' and 'necntx' to 1.
      parameter(nlcntx=3)
      parameter(necntx=3)

c     Set number of pixels in MAS scan lines.
      parameter(npixel = 716)

c     Set the total number of MAS channels.
      parameter(nbands = 50)

c     Define midpoint of scan for sub-satellite point calculations.
      parameter(mdele1 = 357)
      parameter(mdele2 = 358)

c     Set number of bands used as input.
      parameter(inband = 13)

c**********************************************************************

c     Variable declarations.

      character*72 sds_name
      character*50 out_file,in_file,parm(8)

      integer*2 indat(npixel,nbands)

      integer*4 outrec,tp_unit,eco_unit,outf_unit,lst(npixel,nlcntx),
     *          rglst(nlcntx,necntx),
     *          npix,n1sm,n2sm,n3sm,n4sm,pcwatr(npixel,nlcntx),
     *          rgpcwt(nlcntx,necntx),dataid,szaid,vzaid,solirid,
     *          slcid,sltid,slymdid,cwlid,inchan(inband),solazid,
     *          inchns(inband),start(3),stride(3),edge(3),snsazid,
     *          gstart(3),gstride(3),gedge(3),atr_indx,dims(3),
     *          sffattr,sfrattr,sfrdata,sfginfo,dhr2hms,eco_index

      real*4 pxldat(nbands),sza(npixel,nlcntx),
     *       vza(npixel,nlcntx),raz(npixel,nlcntx),
     *       refang(npixel,nlcntx),rdat(npixel,nlcntx,nbands),
     *       pw,avgtherm(nbands),rgdata(nlcntx,necntx,nbands),
     *       rglat(nlcntx,necntx),rglon(nlcntx,necntx),
     *       rgsza(nlcntx,necntx),rgvza(nlcntx,necntx),
     *       rgraz(nlcntx,necntx),rgrfa(nlcntx,necntx),
     *       rlats(npixel,nlcntx),rlons(npixel,nlcntx),
     *       rlat(npixel),rlon(npixel),rsza(npixel),
     *       rvza(npixel),cwl(50),scf(50),solirr(50),
     *       diff(nlcntx*necntx-1,nbands),rsolaz(npixel),
     *       rsnsaz(npixel)

      byte eco_type,testbits(2),bitarray(2,npixel),
     *     ecstypes(npixel,nlcntx),rgeco(nlcntx,necntx)

      logical*4 snglnt,polar,water,land,day,night,visusd,snow,
     *          end_file,shadow,uniform,ap_sea_ice,ap_snow,ice,
     *          coast,desert,vrused,cnlwc

c**********************************************************************

c     Initializations.

      data outrec/1/, bitarray/1432*0/
      data end_file /.false./

c     For WINCE data.
      data inchan /2,3,7,9,10,15,20,30,31,42,45,46,49/
```

```
c     For SUCCESS data.
c     data inchan /1,2,7,9,10,15,20,31,32,42,45,46,49/

      data stride /1,1,1/
      data gstride /1,1,1/

c*********************************************************************

      include 'hdf.inc'

c*********************************************************************

c     Get input from keyboard.
      do num_in = 1,8
        call getarg(num_in,parm(num_in))
      enddo

      read(parm(1),'(a50)') in_file
      read(parm(2),'(a50)') out_file
      read(parm(3),'(i10)') ibl
      read(parm(4),'(i10)') nlin
      read(parm(5),'(i10)') ibele
      read(parm(6),'(i10)') nele
      read(parm(7),'(i10)') eco_index
      read(parm(8),'(f10.5)') pw

      if ((pw.lt.1.) .or. (pw.gt.11.)) then
        write (*,'(''  tri-spectral technique will not be used '')')
      end if

c*********************************************************************

c     Open files and obtain corresponding id/unit numbers.

      call file_open(in_file,out_file,eco_index,if_hdfid,tp_unit,
     *               eco_unit,outf_unit,irt_code)
      if(irt_code .lt. 0) go to 1000

c*********************************************************************

c     Make HDF calls to retrieve all necessary HDF id numbers, data scale
c     factors, channel central wavelengths and spectral solar irradiance.

c     Get IDs.
      call set_HDF(if_hdfid,dataid,latid,lonid,szaid,vzaid,slcid,
     *             sltid,slymdid,cwlid,solirid,solazid,snsazid)

c     Get scale factors for radiance data.
      atr_indx = sffattr(dataid,'scale_factor')
      irt = sfrattr(dataid,atr_indx,scf)

c     Get channel central wavelengths.
      start(1) = 0
      edge(1) = 50
      irt = sfrdata(cwlid,start,stride,edge,cwl)

c     Get spectral solar irradiance for each channel.
      irt = sfrdata(solirid,start,stride,edge,solirr)

c*********************************************************************

c     Get date and time of first scan line in the data.

      start(1) = 0
      edge(1) = 1

c     Year, month, day from input file (yymmdd).
      irt = sfrdata(slymdid,start,stride,edge,idbymd)

c     Time in decimal hours from input file.
      irt = sfrdata(sltid,start,stride,edge,dbtime)

c     Convert decimal hours to hours, minutes, seconds (hhmmss).
      ibhms = dhr2hms(dbtime)

c     Convert date from "yymmdd" to "yyddd".
      iyr = idbymd / 10000
      imo = mod((idbymd/100),(iyr*100))
      ida = mod(idbymd,(iyr*10000+imo*100))
      nda = numday(iyr,imo,ida)
      iyrday = (mod(iyr,100)*1000) + nda
      jday = mod(iyrday,1000)

      write(*,'(1x,''Beginning year,month,day: '',i4,i3,i4)')iyr,imo,ida
      write(*,'(1x,''Beginning time in decimal hours: '',f10.5)') dbtime

c*********************************************************************

c     Get number of scanlines in the file.
      irt = sfginfo(dataid,sds_name,rank,dims,num_type,nattrs)
      maxlin = dims(3)
      write(*,'(1x,''# lines in file: '',i10)') maxlin

c*********************************************************************

c     Define constants based on user inputs.

c     'ibls' is first line # needed
c     'nelin' is last line # needed
c     'ibes' is first element # needed
c     'nelet' is last element # needed on each scan line

      ibls = ibl - ((nlcntx-1) / 2)
      ibes = ibele - ((necntx-1) / 2)
      nelin = ibls + (nlcntx-1) + (nlin-1)
      nelet = ibes + (necntx-1) + (nele-1)

c*********************************************************************

c     Check user input for consistency with data file size.

      call chk_input(ibls,ibes,nelin,nelet,maxlin,npixel,
     *               nlcntx,necntx,irt_code)
      if(irt_code .lt. 0) go to 1000

c*********************************************************************

c     Begin input scan line loop.

c     'mlin' counts # lines processed in current region (context).
c     'start(1)' is beginning pixel to read from scanline (always 0).
c     'edge(1)' is the # of pixels to read from scanline (always 'npixel').
c     'edge(2)' is the # of channels to read (always 1).
c     'edge(3)' is the # scanlines to read (always 1).
c     'stride(1,2,3)' is increment defined above (always 1).
c     'gstart(1)' is beginning pixel to read from geometry (always 0).
c     'gedge(1)' is the # of pixels to read from geometry (always 'npixel').
c     'gedge(2)' is the # geometry lines to read (always 1).
c     'gstride(1,2,3)' is increment defined above (always 1).
c     'line' is current line #.
```

```
        mlin = 0
        start(1) = 0
        edge(1) = npixel
        edge(2) = 1
        edge(3) = 1
        gstart(1) = 0
        gedge(1) = npixel
        gedge(2) = 1

        Do 100 line=ibls,nelin

          if(line .le. maxlin) then

c           Get data from a region encompassing the number of scans in a context
c           ('nlcntx') in the along-track direction and the total number of
c           pixels in the MAS scan (a "data block"). After the first data block
c           has been filled, only one additional scan is needed with each pass
c           through the loop.

            mlin = mlin + 1

c           Read scan line radiance data for input channels.
c           'start(3)' is the scan line number at which to begin.
c           'start(2)'is the channel number at which to begin.

            start(3) = line - 1
            do nc = 1,inband

              ich = inchan(nc)
              start(2) = ich - 1

              irt = sfrdata(dataid,start,stride,edge,indat(1,nc))
c             write(*,'(1x,''data '',i5,i10)') ich,indat(358,nc)

              inchns(nc) = inchan(nc)

            enddo

c           Read latitudes, longitudes, solar zenith angles, viewing zenith
c           angles for current scan line.
c           'gstart(2)' is the scan line number at which to begin.

            gstart(2) = line - 1
            irt = sfrdata(latid,gstart,gstride,gedge,rlat)
            irt = sfrdata(lonid,gstart,gstride,gedge,rlon)
            irt = sfrdata(szaid,gstart,gstride,gedge,rsza)
            irt = sfrdata(vzaid,gstart,gstride,gedge,rvza)
            irt = sfrdata(solazid,gstart,gstride,gedge,rsolaz)
            irt = sfrdata(snsazid,gstart,gstride,gedge,rsnsaz)

c           Get scan line time.
            irt = sfrdata(sltid,line-1,stride,1,sltime)
            islhms = dhr2hms(sltime)
c           If middle scan of the first context, record the time. This is
c           the time of the first output record of the cloud mask.
            ifsc = (nlcntx-1) / 2
            if(line .eq. (ibls+ifsc)) islhms1 = dhr2hms(sltime)

c           Calculate any additional geometry for necessary pixels on the current
c           scan line.
            call get_geo(npixel,nlcntx,mdele1,mdele2,ibes,
     *                   nelet,mlin,rlat,rlon,rsza,rvza,rsolaz,
     *                   rsnsaz,rlats,rlons,sza,vza,raz,refang)

c           Get land/sea tag and ecosystem type for necessary pixels on the
c           current scan line.
            call get_sfc(npixel,nlcntx,ibes,nelet,mlin,rlats,rlons,
     *                   tp_unit,eco_index,eco_unit,ecstypes,lst,pcwatr)

c           Unscale brightness temperature values and convert visible radiances
c           to reflectances for necessary pixels on the current scan line.
            call get_data(npixel,nlcntx,ibes,nelet,mlin,inband,nbands,
     *                    inchns,cwl,indat,sza,vza,scf,solirr,rdat)

            if(mlin .eq. nlcntx) then

c             Data block has been filled. In this version, the context is
c             defined as ('nlcntx' * 'necntx') pixels centered on the current
c             pixel. This box "slides along" the data block as the pixel number
c             is incremented. Therefore, the number of contexts in a scan line
c             is the same as the number of pixels processed ('nele').

              nmcntx = nele

c             'jlin' is the current scan line being processed (middle scan
c             of the current data block).
c             'klin' is the first line in the current data block.

              jlin = line - ((nlcntx-1) / 2)
              klin = mlin - (nlcntx-1)
              if(mod(jlin,100) .eq. 0) then
                write(*,'(1x,''Processing data block at line '',i10)')
     *                jlin
              end if

c             'kele' is first pixel of the scan to be processed.
c             'jele' is first required pixel of the scan.

              kele = ibele
              jele = ibes

c*********************************************************************

c             Loop over the number of contexts in the current data block.

              do 200 nc = 1,nmcntx

c               Get ancillary data for all pixels in the current context (region)
c               and set logical flags.
                call reg_anc(npixel,nlcntx,necntx,jele,klin,rlats,
     *                       rlons,sza,vza,raz,refang,pcwatr,ecstypes,
     *                       rglat,rglon,rgsza,rgvza,rgraz,rgrfa,
     *                       rgpcwt,lst,rglst,rgeco,uniform,cnlwc)

c               Get reflectance and brightness temperature data for all pixels
c               in the current context.
                call reg_data(npixel,nlcntx,necntx,jele,klin,inband,
     *                        nbands,rdat,rgdata)

c               Get reflectance and BT values for current pixel.
                call get_pxldat(nlcntx,necntx,inband,nbands,rgdata,
     *                          pxldat)

c               Quick averaging of MAS BT's over context to reduce noise.
                call atherm(nlcntx,necntx,inband,nbands,inchns,rgdata,
     *                      avgtherm)

c               Get ancillary data and set logical flags for the current
c               pixel (middle pixel of current context).
```

```
            call pxl_anc(nlcntx,necntx,rgvza,rgsza,rgrfa,rgpcwt,rgeco,
     *                   rglst,rglat,pxvza,pxrfa,pxpcwt,eco_type,
     *                   day,night,land,water,polar,snglnt,cnlwc,
     *                   visusd,vrused,ap_sea_ice,ap_snow,snow,ice,
     *                   coast,desert,nbands,pxldat,avgtherm,pxsza,
     *                   jday)
c
c           Initialize (clear) output "bit" array.
            do iwd = 1,2
              testbits(iwd) = 0
            enddo

c           Set "Additional Information" and "spare" bits (value=1).
            call set_bit(testbits,7)
            call set_bit(testbits,8)
            call set_bit(testbits,9)
            call set_bit(testbits,15)

c*********************************************************************

c           Begin tests for cloud mask determination.

            if (polar .and. night) then

c             Polar nighttime processing.

              call polar_nite(pxldat,pxvza,eco_type,land,ice,snow,
     *                        uniform,pw,nlcntx,necntx,inband,nbands,
     *                        rgdata,avgtherm,diff,testbits,
     *                        confdnc)

            else if(coast) then

c             Coastal region.

              if(day) then
c               Daytime processing.
                call coast_day(pxldat,pxvza,visusd,eco_type,
     *                         desert,snow,nlcntx,necntx,nbands,
     *                         avgtherm,testbits,confdnc)

              else
c               Nighttime processing.
                call coast_nite(pxldat,pxvza,eco_type,snow,nlcntx,
     *                          necntx,nbands,avgtherm,testbits,
     *                          confdnc)

              end if

            else if (water) then

c             Primarily water surface.

              if (day) then
c               Daytime processing.
                call water_day(pxldat,pxvza,pxrfa,snglnt,visusd,
     *                         ice,uniform,eco_type,pw,nlcntx,
     *                         necntx,inband,nbands,rgdata,
     *                         avgtherm,diff,testbits,confdnc)
              else
c               Nighttime processing.
                call water_nite(pxldat,pxvza,eco_type,ice,uniform,
     *                          pw,nlcntx,necntx,inband,nbands,
     *                          rgdata,avgtherm,diff,
     *                          testbits,confdnc)
              end if

            else if(land) then

c             Process as land surface.

              if (day) then
c               Daytime processing.
                call land_day(pxldat,pxvza,visusd,vrused,
     *                        eco_type,desert,snow,nlcntx,necntx,
     *                        nbands,avgtherm,testbits,confdnc)
              else
c               Nighttime processing.
                call land_nite(pxldat,pxvza,eco_type,snow,nlcntx,
     *                         necntx,nbands,avgtherm,testbits,
     *                         confdnc)
              end if

            end if

c*********************************************************************

c           Test for shadows, if necessary.
            if(land.and.day.and..not.snow.and.confdnc.ge.0.66) then
              call shadows(nbands,pxldat,shadow,testbits)
            end if

c           Test for possible non-cloud obstruction.
            if(land .and. day .and. .not. polar .and. .not. snow) then
              call noncld_obs_chk(nbands,pxldat,pxsza,testbits)
            end if

c           Set bits which indicate processing path through algorithm.
            call proc_path(water,land,coast,desert,day,snglnt,
     *                     snow,ice,testbits)

c           Get cloud mask statistics.
            call get_stats(confdnc,npix,n1sm,n2sm,n3sm,n4sm)

c           Set cloud mask quality bit flags.
            call set_confdnc(confdnc,testbits)

            do 250 ibyte = 1,2
              bitarray(ibyte,kele) = testbits(ibyte)
 250        continue

c           Increment counters.
            kele = kele + 1
            jele = jele + 1

 200      continue

c*********************************************************************

c         Write bit flags to output file for current scan line.

          outrec = outrec + 1
          call write_bits(outf_unit,bitarray,iyrday,ibhms,islhms1,
     *                    outrec,end_file)

c*********************************************************************

c         If not processing in single-line mode, re-buffer data in order
c         to add the next scan line and make a new data block.
```

```
          if(nlcntx .gt. 1) then
            do 400 il = 2,nlcntx
              do 500 ie = ibes,nelet
                rlats(ie,il-1) = rlats(ie,il)
                rlons(ie,il-1) = rlons(ie,il)
                sza (ie,il-1) = sza (ie,il)
                vza (ie,il-1) = vza (ie,il)
                raz (ie,il-1) = raz (ie,il)
                refang(ie,il-1) = refang(ie,il)
                pcwatr(ie,il-1) = pcwatr(ie,il)
                ecstypes(ie,il-1) = ecstypes(ie,il)
                lst (ie,il-1) = lst (ie,il)
                do 600 k = 1,inband
                  rdat(ie,il-1,k) = rdat(ie,il,k)
 600            continue
 500          continue
 400        continue
          end if

          mlin = nlcntx - 1

        endif

      end if

 100  continue

c***********************************************************************

c     Write total number records output to the header record of the output
c     file.

      end_file = .true.
      call write_bits(outf_unit,bitarray,iyrday,ibhms,islhms1,outrec,
     *                end_file)

c***********************************************************************

c     Write out stats.

      call stats_out(npix,n1sm,n2sm,n3sm,n4sm)

c***********************************************************************

 1000 continue

c     Close all files.

      call file_close(if_hdfid,tp_unit,eco_unit,
     *                outf_unit)

c***********************************************************************

      write(*,'(1x,''MASGLIMSK finished'')')

      stop
      end
```

```fortran
      subroutine file_open(in_file,out_file,eco_index,if_hdfid,
     *                     tp_unit,eco_unit,outf_unit,irt_code)

c     Routine for opening all files needed for processing.
c     Returns necessary unit numbers to calling routine.

      character*50,topog,eco_file1,eco_file2,in_file,out_file
      integer*4 tp_unit,eco_unit,outf_unit,tp_un,eco_un,
     *          outf_un,sfstart,eco_index

c****************************************************************

c     Names of topography and ecosystem files.
      data topog /'lst1km.v3'/
      data eco_file1 /'naogel_01g.img'/
      data eco_file2 /'ecosystem.img'/

c****************************************************************

c     Define unit numbers for ancillary and output files.
      data tp_un /30/ , eco_un /80/ , outf_un /70/

      include 'hdf.inc'

c     Initialize return code.
      irt_code = 0

c     Open files.

c     Topography
      open (tp_un,file=topog,status='old',iostat=irt,
     *      access='direct',form='unformatted',recl=11)
      if(irt .ne. 0) then
        write(*,'(1x,''Cannot open topography file on unit '',
     *           i5)') tp_un
        irt_code = irt
      end if

c     Ecosystems
      if(eco_index .eq. 1) then
c       North American 1 km
        open(eco_un,file=eco_file1,status='old',iostat=irt,
     *       access='direct',form='unformatted',recl=11329)
      else
c       Global 10 minute
        open(eco_un,file=eco_file2,status='old',iostat=irt,
     *       access='direct',form='unformatted',recl=2)
      end if
      if(irt .ne. 0) then
        write(*,'(1x,''Cannot open ecosystem file on unit '',
     *           i5)') eco_un
        irt_code = irt
      end if

c     Output binary cloud mask file (bit flags).
      write(*,'(1x,''Opening output file '',a50)') out_file
      open(outf_un,file=out_file,status='unknown',iostat=irt,
     *     access='direct',form='unformatted',recl=1432)
      if(irt .ne. 0) then
        write(*,'(1x,''Cannot open output file on unit '',
     *           i5)') outf_un
        irt_code = irt
      end if

c     Open input HDF data file and initialize HDF software.

      if_hdfid = sfstart(in_file,DFACC_READ)

      tp_unit = tp_un
      eco_unit = eco_un
      outf_unit = outf_un

      if(irt_code .ne. 0) irt_code = -1

      return
      end
```

```
      subroutine set_HDF(if_hdfid,dataid,latid,lonid,szaid,vzaid,
     *                   slcid,sltid,slymdid,cwlid,solirid,solazid,
     *                   snsazid)

c     Routine which retrieves HDF indices for reading the input HDF file.

      integer*4 dataid,latid,lonid,szaid,vzaid,slcid,sltid,slymdid,
     *          cwlid,solirid,solazid,snsazid,sds_indx,sfn2index,
     *          sfselect

      include 'hdf.inc'

      sds_indx = sfn2index(if_hdfid,'CalibratedData')
      dataid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'Central100%ResponseWavelength')
      cwlid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'PixelLongitude')
      lonid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'PixelLatitude')
      latid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'SolarZenithAngle')
      szaid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'SensorZenithAngle')
      vzaid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'ScanLineCounter')
      slcid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'ScanlineTime')
      sltid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'YearMonthDay')
      slymdid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'SolarSpectralIrradiance')
      solirid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'SolarAzimuthAngle')
      solazid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'SensorAzimuthAngle')
      snsazid = sfselect(if_hdfid,sds_indx)

      return
      end
```

```
      subroutine chk_input(ibls,ibes,nelin,nelet,maxlin,maxele,
     *                     nlcntx,necntx,irt_code)

c     Check consistency of input information with number of scanlines
c     and pixels available.

c     Initialize return code
      irt_code = 0

      if(ibls .lt. 1 .or. nelin .gt. maxlin) then
         write(*,'(1x,''Invalid starting line or too many lines'')')
         irt_code = -1
      end if

      if(ibes .lt. 1 .or. nelet .gt. maxele) then
         write(*,'(1x,''Invalid starting pixel or too many pixels'')')
         irt_code = -2
      end if

      return
      end
```

```
      integer function dhr2hms(dechrs)

c     Function which converts time in decimal hours to hms format.

      real*4 dechrs

      ih = dechrs

      a = dechrs - float(ih)
      b = a * 60.0
      im = b

      c = b - float(im)
      is = nint(c * 60.0 )

      if(is .eq. 60) then
        im = im + 1
        is = 0
      end if

      if(im .eq. 60) then
        ih = ih + 1
        im = 0
      end if

      dhr2hms = ih*10000 + im*100 + is

      return
      end
```

```
      function numday(iyr,imo,ida)

c     Calculates numeric day of the year given month and day of year.

      integer*4 idays(12),ildays(12)

      data idays /0,31,59,90,120,151,181,212,243,273,304,334/
      data ildays /0,31,60,91,121,152,182,213,244,274,305,335/

      if(mod(iyr,4) .ne. 0) then
        numday = idays(imo) + ida
      else
        numday = ildays(imo) + ida
      end if

      return
      end
```

```
      subroutine get_geo(npixel,nlcntx,mdele1,mdele2,spx,
     *                   epx,mlin,rlat,rlon,rsza,rvza,rsolaz,
     *                   rsnsaz,rlats,rlons,sza,vza,raz,
     *                   refang)

c     Routine which returns relative azimuth angle and reflectance
c     angle for the specified MAS FOVs on a scan line.  Also stores
c     all necessary geometric data in arrays conforming to a "data
c     block" (geometry for all necessary pixels in each of 'nlcntx'
c     scan lines).  See main program.

      parameter(pi = 3.14159)
      parameter(dtr = pi/180.0)
      parameter(rtd = 180.0/pi)

      integer*4 spx,epx
      real*4 sza(npixel,nlcntx),vza(npixel,nlcntx),
     *       raz(npixel,nlcntx),refang(npixel,nlcntx),
     *       rlats(npixel,nlcntx),rlons(npixel,nlcntx),
     *       rlat(npixel),rlon(npixel),rsza(npixel),rvza(npixel),
     *       rsolaz(npixel),rsnsaz(npixel)

c     Get sub-satellite latitude and longitude.

      slat1 = rlat(mdele1)
      slon1 = rlon(mdele1)
      slat2 = rlat(mdele2)
      slon2 = rlon(mdele2)
      sslat = (slat1 + slat2) / 2.0
      sslon = (slon1 + slon2) / 2.0

c     Loop through the needed FOVs.

      do npx = spx,epx

c       Get lat, lon, solar azimuth, sensor azimuth for current pixel.
        plat = rlat(npx)
        plon = rlon(npx)
        psolaz = rsolaz(npx)
        psnsaz = rsnsaz(npx)

c       Get relative azimuth angle (as defined by ERBE).
        rlaz = abs(180.0 - (abs(psnsaz - psolaz)))

c       Get solar reflectance angle for sun glint determination.

        vzar = rvza(npx)*dtr
        szar = rsza(npx)*dtr
        razr = rlaz*dtr
        cossna = sin(vzar)*sin(szar)*cos(razr) +
     *           cos(vzar)*cos(szar)
        rfa = acos(cossna) * rtd

        sza(npx,mlin) = rsza(npx)
        vza(npx,mlin) = rvza(npx)
        raz(npx,mlin) = rlaz
        refang(npx,mlin) = rfa
        rlats(npx,mlin) = plat
        rlons(npx,mlin) = plon

      enddo

      return
      end
```

```
      subroutine get_sfc(npixel,nlcntx,spx,epx,mlin,rlats,
     *                   rlons,tp_unit,eco_index,eco_unit,
     *                   ecstypes,lst,pcwatr)

c     Routine which returns land/sea tags and ecosystem types for
c     the specified pixels on a scan line. Also stores surface
c     data in arrays conforming to a "data block" (see main program).

      integer*4 tp_unit,eco_unit,pcwatr(npixel,nlcntx),spx,epx,
     *          eco_index,lst(npixel,nlcntx)
      real*4 rlats(npixel,nlcntx),rlons(npixel,nlcntx)
      real*8 dlat,dlon,line,samp,dlat1,dlon2
      byte map(11),ecorec_g(2),ecorec_1(11329),ecstypes(npixel,nlcntx)

      save nrec

      data nrec /0/

c     Loop through pixels of interest.

      do npx = spx,epx

c       Get latitude and longitude of current pixel.
        rlat = rlats(npx,mlin)
        rlon = rlons(npx,mlin)

c       Get land/sea flag (use 1 km map).
c
c       Get map coordinates corresponding to the lat, lon position.
c
        dlat = rlat
        dlon = rlon
        dlat1 = rlat
        dlon2 = rlon
        call getcoord(dlat,dlon,line,samp)
        irec = nint(line)
        ismp = nint(samp)
c       Read land/sea tag file.
        ibbyte = ((irec-1)*40031) + (ismp-1)

        newlin = ibbyte / 11 + 1
        newele = mod(ibbyte,11)
        if(mod(ibbyte,11) .eq. 0) then
          newlin = newlin - 1
          newele = 11
        end if
        read(tp_unit,rec=newlin) map
        lsf = map(newele)
        lst(npx,mlin) = lsf
        if(lsf .eq. 1 .or. lsf .eq. 4) then
          pcwatr(npx,mlin) = 0
        else if(lsf .eq. 2) then
          pcwatr(npx,mlin) = 50
        else
          pcwatr(npx,mlin) = 100
        end if

c       Get ecosystem type.

        if(eco_index .ne. 1) then

c         Use global 10 minute file.

c         Calculate ecosystem file record number using standard
c         longitude.
          lat_indx = nint((90.0 - rlat) * 6.0) + 1
          lon_indx = nint((rlon+180.0) * 6.0) + 1
          if(lat_indx .gt. 1080) lat_indx = 1080
          if(lon_indx .gt. 2160) lon_indx = 2160
          norec = ((lat_indx-1) * 2160) + lon_indx
          newlin = norec / 2 + 1
          newele = mod(norec,2)
          if(mod(norec,2) .eq. 0) then
            newlin = newlin - 1
            newele = 2
          end if
c         Read ecosystem file.
          read(eco_unit,rec=newlin) ecorec_g
          ecstypes(npx,mlin) = ecorec_g(newele)

        else

c         Get type from 1 km resolution N. American file.

          call nacoord(dlat1,dlon2,line,samp)
          irec = nint(line)
          ismp = nint(samp)
          if(irec .ne. nrec) then
            read(eco_unit,rec=irec) ecorec_1
          end if
          ecstypes(npx,mlin) = ecorec_1(ismp)
          nrec = irec

        end if

      enddo

      return
      end
```

```fortran
      subroutine get_data(npixel,nlcntx,spx,epx,mlin,inband,nbands,
     *                    inchns,cwl,indat,sza,vza,scf,
     *                    solirr,rdat)

c     Converts thermal channel radiances to brightness temperatures and
c     visible radiances to reflectances for the specified MAS
c     pixels on a scan line. Stores output values in arrays conforming
c     to a "data block" (see main program).

      integer*2 indat(npixel,nbands)
      integer*4 inchns(inband),spx,epx
      real*4 sza(npixel,nlcntx),vza(npixel,nlcntx),
     *       rdat(npixel,nlcntx,nbands),scf(nbands),cwl(nbands),
     *       solirr(nbands)

      parameter(pi = 3.14159)
      parameter(dtr = pi/180.0)

c     Process scan line input data.

      do k = 1,inband

        n = inchns(k)

        do j = spx,epx

          szen = sza(j,mlin)
          vzen = vza(j,mlin)

          if(indat(j,k).ge.0.and.indat(j,k).ne.32767)then
            if(n .le. 25) then
              if (szen .lt. 85.0) then
                rad = indat(j,k) * scf(n)
                csza = cos(dtr*szen)
                refstor = (rad*pi) / (solirr(n)*csza)
                rdat(j,mlin,k) = refstor * 100.0
              else
                rdat(j,mlin,k) = 0.0
              endif
            else
              rad = indat(j,k) * scf(n)
              rdat(j,mlin,k) = bright50(n,cwl(n),rad)
            end if
          else
            rdat(j,mlin,k) = 32767.0
          end if

        enddo

      enddo


      return
      end
```

```
      subroutine reg_anc(npixel,nlcntx,necntx,jele,klin,
     *                   rlats,rlons,sza,vza,raz,refang,pcwatr,
     *                   ecstypes,rglat,rglon,rgsza,rgvza,rgraz,
     *                   rgrfa,rgpcwt,lst,rglst,rgeco,uniform,
     *                   cnlwc)

c     Routine which fills context (regional) ancillary data arrays.
c     Data is obtained from data block arrays.

      integer*4 pcwatr(npixel,nlcntx),ref_eco,lst(npixel,nlcntx),
     *          rglst(nlcntx,necntx),rgpcwt(nlcntx,necntx),pxpcwt

      real*4 sza(npixel,nlcntx),vza(npixel,nlcntx),
     *       raz(npixel,nlcntx),refang(npixel,nlcntx),
     *       rglat(nlcntx,necntx),rglon(nlcntx,necntx),
     *       rgsza(nlcntx,necntx),rgvza(nlcntx,necntx),
     *       rgraz(nlcntx,necntx),rgrfa(nlcntx,necntx),
     *       rlats(npixel,nlcntx),rlons(npixel,nlcntx)

      logical*4 uniform,ref_flag(4),cnlwc

      byte ecstypes(npixel,nlcntx),rgeco(nlcntx,necntx)

c     Fill regional context arrays.

      je = jele - 1
      kl = klin - 1
      do i = 1,nlcntx
        do j = 1,necntx
          rglat(i,j) = rlats(je+j,kl+i)
          rglon(i,j) = rlons(je+j,kl+i)
          rgsza(i,j) = sza(je+j,kl+i)
          rgvza(i,j) = vza(je+j,kl+i)
          rgraz(i,j) = raz(je+j,kl+i)
          rgrfa(i,j) = refang(je+j,kl+i)
          rgpcwt(i,j) = pcwatr(je+j,kl+i)
          rgeco(i,j) = ecstypes(je+j,kl+i)
          rglst(i,j) = lst(je+j,kl+i)
        enddo
      enddo

c     Check for regional uniformity. If any pixel in the current region
c     is inconsistent with the rest, then the region is non-uniform.
c     Categories are polar/non-polar, day/night, sun-glint, ecosystem,
c     and land/water.

      do i = 1,4
        ref_flag(i) = .true.
      enddo

      do i = 1,nlcntx
        do j = 1,necntx
          if(i .eq. 1 .and. j .eq. 1) then
            if(abs(rglat(i,j)) .le. 60.0) ref_flag(1) = .false.
            if(rgsza(i,j) .ge. 85.0) ref_flag(2) = .false.
            if(rgrfa(i,j) .gt. 36.0) ref_flag(3) = .false.
            if(rgpcwt(i,j) .ne. 100) ref_flag(4) = .false.
            ref_eco = rgeco(i,j)
          else
            if((abs(rglat(i,j)) .gt. 60.0) .neqv. ref_flag(1)) go to 100
            if((rgsza(i,j) .lt. 85.0) .neqv. ref_flag(2)) go to 100
            if((rgrfa(i,j) .le. 36.0) .neqv. ref_flag(3)) go to 100
            if((rgpcwt(i,j) .eq. 100) .neqv. ref_flag(4)) go to 100
            if(ref_eco .ne. rgeco(i,j)) go to 100
          end if
        enddo
      enddo

      uniform = .true.
      go to 200

 100  uniform = .false.

 200  continue

c     Determine if current pixel is in a coastal region.

      nland = 0
      nwater = 0
      ncoast = 0
      itotal = nlcntx * necntx

      do i = 1,nlcntx
        do j = 1,necntx

c         Check land/sea tag (% coverage of water in pixel).
          pxpcwt = rgpcwt(i,j)
          if(pxpcwt .eq. 100) then
            nwater = nwater + 1
          else if(pxpcwt .eq. 50) then
            ncoast = ncoast + 1
          else if(pxpcwt .eq. 0) then
            nland = nland + 1
          end if

        enddo
      enddo

      if( (nwater+ncoast) .eq. itotal) then
        cnlwc = .true.
      else if( (nland+ncoast) .eq. itotal) then
        cnlwc = .true.
      else
        cnlwc = .false.
      end if

      return
      end
```

```
      subroutine reg_data(npixel,nlcntx,necntx,jele,klin,
     *                    inband,nbands,rdat,rgdata)

c     Routine for getting regional reflectance and thermal
c     data from "data block" array.

      real*4 rdat(npixel,nlcntx,nbands),
     *       rgdata(nlcntx,necntx,nbands)

c     Fill regional context arrays.

      je = jele - 1
      kl = klin - 1
      do k = 1,inband
        do j = 1,nlcntx
          do i = 1,necntx
            rgdata(j,i,k) = rdat(je+i,kl+j,k)
          enddo
        enddo
      enddo

      return
      end
```

```
      subroutine get_pxldat(nlcntx,necntx,inband,nbands,
     *                      rgdata,pxldat)

c     Returns array of reflectances/brightness temperatures for
c     the current pixel.

      real*4 pxldat(nbands),rgdata(nlcntx,necntx,nbands)

      ie = necntx - ((necntx-1) / 2)
      il = nlcntx - ((nlcntx-1) / 2)

      do k = 1,inband
        pxldat(k) = rgdata(il,ie,k)
      enddo

      return
      end
```

```
      subroutine atherm(nlcntx,necntx,inband,nbands,inchns,rgdata,
     *                  avgtherm)

c     Computes the average brightness temperature over the
c     given box size (context) to reduce the noise.

      integer*4 inchns(inband)
      real*4 avgtherm(nbands),rgdata(nlcntx,necntx,nbands)

      do 100 k = 1,inband

        n = inchns(k)

        if(n .gt. 25) then

          knt = 0
          sum = 0.0

          do 200 i = 1,nlcntx

            do 300 j = 1,necntx

              if(rgdata(i,j,k) .ne. 32767.0) then
                knt = knt + 1
                sum = sum + rgdata(i,j,k)
              end if

300         continue

200       continue

          if(knt .gt. 0) then
            avgtherm(k) = sum / float(knt)
          else
            avgtherm(k) = 32767.0
          end if

        end if

100   continue

      return
      end
```

```fortran
      subroutine pxl_anc(nlcntx,necntx,rgvza,rgsza,rgrfa,rgpcwt,rgeco,
     *                   rglst,rglat,pxvza,pxrfa,pxpcwt,eco_type,
     *                   day,night,land,water,polar,snglnt,cnlwc,
     *                   visusd,vrused,ap_sea_ice,ap_snow,snow,ice,
     *                   coast,desert,nbands,pxldat,avgtherm,pxsza,jday)

c     Returns necessary ancillary data for a given pixel and
c     any logical flags needed for further processing.

      integer*4 rgpcwt(nlcntx,necntx),rglst(nlcntx,necntx),pxlst
      real*4 pxvza,pxrfa,pxpcwt,rgvza(nlcntx,necntx),
     *       rgrfa(nlcntx,necntx),rgsza(nlcntx,necntx),
     *       rglat(nlcntx,necntx),pxldat(nbands),
     *       avgtherm(nbands)
      logical*4 snglnt,visusd,water,land,day,night,polar,snow,
     *          ice,ap_sea_ice,ap_snow,dh_snow,desert,vrused,cnlwc,
     *          coast
      byte rgeco(nlcntx,necntx),eco_type

      ie = necntx - ((necntx-1) / 2)
      il = nlcntx - ((nlcntx-1) / 2)

      pxvza = rgvza(il,ie)
      pxsza = rgsza(il,ie)
      pxrfa = rgrfa(il,ie)
      pxlst = rglst(il,ie)
      pxpcwt = float(rgpcwt(il,ie))
      eco_type = rgeco(il,ie)
      rlat = rglat(il,ie)

c     Determine if current pixel is in geometric sun-glint region.
      if(pxrfa .le. 36.0) then
         snglnt = .true.
      else
         snglnt = .false.
      end if
c     Determine if current pixel will be processed as day or night.
      if(pxsza .lt. 85.0) then
         day = .true.
         visusd = .true.
         night = .false.
      else
         night = .true.
         day = .false.
         visusd = .false.
      end if

c     Determine if current pixel will be processed as land or water
c     (ocean).
      if(pxpcwt .ge. 90.0) then
         water = .true.
         land = .false.
      else
         land = .true.
         water = .false.
      end if

c     Determine if current pixel is in a polar region.
      if(rlat .gt. 60.0 .or. rlat .lt. -60.0) then
         polar = .true.
      else
         polar = .false.
      end if

c     Determine whether or not current pixel will be processed as
c     desert.
      if(eco_type .eq. 8 .or. eco_type .eq. 46 .or.
     *   eco_type .eq. 50 .or. eco_type .eq. 51 .or.
     *   eco_type .eq. 59 .or. eco_type .eq. 71 .or.
     *   eco_type .eq. 11) then
        desert = .true.
      else
        desert = .false.
      end if

c     Determine whether or not visible ratio test may be used over
c     land surfaces.
      if(eco_type .eq. 2 .or. eco_type .eq. 8 .or.
     *   eco_type .eq. 11 .or. eco_type .eq. 40 .or.
     *   eco_type .eq. 41 .or. eco_type .eq. 46 .or.
     *   eco_type .eq. 51 .or. eco_type .eq. 52 .or.
     *   eco_type .eq. 59 .or. eco_type .eq. 71 .or.
     *   eco_type .eq. 50) then
        vrused = .false.
      else
        vrused = .true.
      end if

c     Determine whether or not the current pixel will be processed as
c     coastal.
      if( (pxlst .eq. 2) .or. (.not. cnlwc) ) then
        coast = .true.
      else
        coast = .false.
      end if

c     In the future, we will have a priori information about surface
c     snow and sea ice cover. Set flags here, for now.
      ap_sea_ice = .false.
      ap_snow = .false.

c     Set flags indicating presence of surface snow and ice.

c     Initialize to "false".
      snow = .false.
      ice = .false.

      if(day) then
c       Run Bryan Baum's snow algorithm.
        call snowb(nbands,pxldat,rlat,pxsza,jday,avgtherm,dh_snow)
        if(land .and. dh_snow) snow = .true.
        if(water .and. dh_snow) ice = .true.
      else
c       Use a priori information.
        if(land .and. ap_snow) snow = .true.
        if(water .and. ap_sea_ice) ice = .true.
      end if

      return
      end
```

```fortran
      subroutine polar_nite(pxldat,vza,eco_type,land,ice,snow,
     *                      uniform,pw,nlcntx,necntx,inband,nbands,
     *                      rgdata,avgtherm,diff,testbits,
     *                      confdnc)

c     Routine for providing conditional input parameters pertaining
c     to polar nighttime processing.

      integer*4 varslt
      real*4 pxldat(nbands),vza,confdnc,avgtherm(nbands),pw,
     *       rgdata(nlcntx,necntx,nbands),
     *       diff(nlcntx*necntx-1,nbands)
      logical*4 land,snow,ice,uniform
      byte eco_type,testbits(2)

      if(ice .or. snow) then

        call PolarNite_snow(nbands,pxldat,vza,eco_type,
     *                      avgtherm,testbits,confdnc)

      else if(land) then

        call LandNite_type1(nbands,pxldat,avgtherm,vza,eco_type,
     *                      testbits,confdnc)

      else

        call ocean_nite(nbands,pxldat,avgtherm,vza,pw,testbits,
     *                  confdnc)

c       If confidence is still uncertain, apply the spatial variability
c       test. Also check for regional scene uniformity (see reg_anc.f).

        if(uniform .and. confdnc.le.0.95 .and. confdnc.gt.0.34)then

c         Get brightness temperature differences between pixel
c         of interest and the ones surrounding it.
          call get_regdif(nlcntx,necntx,inband,nbands,
     *                    rgdata,diff)

c         Check variation in the region.
          call spatial_var(nbands,nlcntx,necntx,diff,varslt)

c         Bump up the confidence if spatial variability test showed
c         uniform
          if ((varslt .eq. 1) .and. (confdnc .gt. .66)) then
            confdnc = 0.96
          else if ((varslt .eq. 1) .and. (confdnc .le. 0.66)) then
            confdnc = 0.67
          endif

        end if

      end if

      return
      end
```

```
      subroutine coast_day(pxldat,vza,visusd,eco_type,
     *                     desert,snow,nlcntx,necntx,
     *                     nbands,avgtherm,testbits,confdnc)

c     Routine for setting appropriate flags and processing path
c     for daytime observations over coastal areas.

      real*4 pxldat(nbands),vza,confdnc,avgtherm(nbands)
      logical*4 visusd,snow,desert
      byte eco_type,testbits(2)

      if(snow) then

c       Use tests for snow covered conditions.
        call PolarDay_snow(nbands,pxldat,vza,visusd,eco_type,
     *                     avgtherm,testbits,confdnc)

      else if (desert) then

c       Desert or semi-desert ecosystems.
        call DesertDay_c(nbands,pxldat,vza,visusd,avgtherm,
     *                   testbits,confdnc)

      else

c       Use default land tests.
        call LandDay_c(nbands,pxldat,avgtherm,vza,visusd,
     *                 vrused,eco_type,testbits,confdnc)

      end if

      return
      end
```

```
      subroutine coast_nite(pxldat,vza,eco_type,snow,nlcntx,
     *                      necntx,nbands,avgtherm,testbits,
     *                      confdnc)

c     Routine for setting appropriate flags and processing path
c     for nighttime observations over coastal regions.

      real*4 pxldat(nbands),vza,confdnc,avgtherm(nbands)
      logical*4 snow
      byte eco_type,testbits(2)

      if(snow) then

        call PolarNite_snow(nbands,pxldat,vza,eco_type,avgtherm,
     *                      testbits,confdnc)

      else

        call LandNite_type1(nbands,pxldat,avgtherm,vza,eco_type,
     *                      testbits,confdnc)

      end if

      return
      end
```

```fortran
      subroutine water_day(pxldat,vza,pxrfa,snglnt,visusd,ice,
     *                     uniform,eco_type,pw,nlcntx,necntx,
     *                     inband,nbands,rgdata,avgtherm,
     *                     diff,testbits,confdnc)

c     Routine for setting appropriate flags and processing path
c     for daytime observations over water surfaces.

      integer*4 varslt
      real*4 pxldat(nbands),vza,confdnc,pw,avgtherm(nbands),
     *       rgdata(nlcntx,necntx,nbands),
     *       diff(nbands,nlcntx*necntx-1)
      logical*4 visusd,snglnt,uniform,ice
      byte eco_type,testbits(2)

      if(ice) then

        call PolarDay_snow(nbands,pxldat,vza,visusd,eco_type,avgtherm,
     *                     testbits,confdnc)

      else

        call ocean_day(nbands,pxldat,avgtherm,vza,pxrfa,snglnt,visusd,
     *                 eco_type,pw,testbits,confdnc)

c       If confidence is uncertain, apply the spatial variability test.
c       Also, make sure that the scene has uniform surface characteristics
c       (see reg_anc.f).

        if(uniform .and. confdnc.le.0.95 .and. confdnc.gt.0.34) then

c         Get brightness temperature differences between pixel
c         of interest and the ones surrounding it.
          call get_regdif(nlcntx,necntx,inband,nbands,rgdata,
     *                    diff)

c         Check variation in the region.
          call spatial_var(nbands,nlcntx,necntx,diff,varslt)

c         Bump up the confidence if spatial variability test showed
c         uniform
          if ((varslt .eq. 1) .and. (confdnc .gt. .66)) then
            confdnc = 0.96
          else if ((varslt .eq. 1) .and. (confdnc .le. 0.66)) then
            confdnc = 0.67
          endif

        end if

      end if

      return
      end
```

```
      subroutine water_nite(pxldat,vza,eco_type,ice,uniform,pw,
     *                      nlcntx,necntx,inband,nbands,
     *                      rgdata,avgtherm,diff,testbits,confdnc)


c     Routine for setting appropriate flags and processing path
c     for nighttime observations over water surfaces.

      integer*4 varslt
      real*4 pxldat(nbands),vza,confdnc,pw,avgtherm(nbands),
     *       rgdata(nlcntx,necntx,nbands),
     *       diff(nbands,nlcntx*necntx-1)
      logical*4 ice,uniform
      byte eco_type,testbits(2)

      if(ice) then

        call PolarNite_snow(nbands,pxldat,vza,eco_type,avgtherm,
     *                      testbits,confdnc)

      else

        call ocean_nite(nbands,pxldat,avgtherm,vza,pw,testbits,
     *                  confdnc)

c       If confidence is still uncertain, apply the spatial variability
c       test. Also check for regional scene uniformity (see reg_anc.f).

        if(uniform .and. confdnc.le.0.95 .and. confdnc.gt.0.34)then

c         Get brightness temperature differences between pixel
c         of interest and the ones surrounding it.
          call get_regdif(nlcntx,necntx,inband,nbands,rgdata,
     *                    diff)

c         Check variation in the region.
          call spatial_var(nbands,nlcntx,necntx,diff,varslt)

c         Bump up the confidence if spatial variability test showed
c         uniform
          if ((varslt .eq. 1) .and. (confdnc .gt. .66)) then
            confdnc = 0.96
          else if ((varslt .eq. 1) .and. (confdnc .le. 0.66)) then
            confdnc = 0.67
          endif

        end if

      end if

      return
      end
```

```
      subroutine land_day(pxldat,vza,visusd,vrused,eco_type,
     *                    desert,snow,nlcntx,necntx,nbands,
     *                    avgtherm,testbits,confdnc)

c     Routine for setting appropriate flags and processing path
c     for daytime observations over land surfaces.

      real*4 pxldat(nbands),vza,confdnc,avgtherm(nbands)
      logical*4 visusd,snow,desert,vrused
      byte eco_type,testbits(2)

      if(snow) then

c       Use tests for snow covered conditions.
        call PolarDay_snow(nbands,pxldat,vza,visusd,eco_type,
     *                     avgtherm,testbits,confdnc)

      else if (desert) then

c       Desert or semi-desert ecosystems.
        call DesertDay(nbands,pxldat,vza,visusd,avgtherm,
     *                 testbits,confdnc)

      else

c       Use default land tests.
        call LandDay_type1(nbands,pxldat,avgtherm,vza,visusd,
     *                     vrused,eco_type,testbits,confdnc)

      end if

      return
      end
```

```
      subroutine land_nite(pxldat,vza,eco_type,snow,nlcntx,
     *                     necntx,nbands,avgtherm,testbits,
     *                     confdnc)

c     Routine for setting appropriate flags and processing path
c     for nighttime observations over land surfaces.

      real*4 pxldat(nbands),vza,confdnc,avgtherm(nbands)
      logical*4 snow
      byte eco_type,testbits(2)

      if(snow) then

        call PolarNite_snow(nbands,pxldat,vza,eco_type,avgtherm,
     *                      testbits,confdnc)

      else

        call LandNite_type1(nbands,pxldat,avgtherm,vza,eco_type,
     *                      testbits,confdnc)

      end if

      return
      end
```

```
      subroutine shadows(nbands,pxldat,shadow,testbits)

c F77 ***********************************************************
c ... determines the presence of shadows and sets the
c ... appropriate bit flag.
c
c Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c Output Parameters:
c shadow        logical variable indicating shadow is present if .true.
c testbits      two word 1-byte array containing test results
c
c END**************************************************************
c
      include 'thresholds.inc'

c ... scalar arguments ..
      integer nbands
      logical*4 shadow
c ...
c ... array arguments ..
      real pxldat(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real masv88,masv66,masv945,masv162
c ...
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv945 = pxldat(4)

c     Reflectance at 0.945 um must be ge 12% and "visible ratio" gt the
c     ocean threshold or else we're seeing a shadow.  The test on the
c     ratio is to assure that the scene is not clear-sky conditions over
c     a sub-grid scale water body.

      vrat = masv88 / masv66

      if(masv945 .lt. 12.0 .and. vrat .gt. dovratlo(2)) then
        shadow = .true.
        call clear_bit(testbits,9)
      else
        shadow = .false.
      end if

      return
      end
```

```
      subroutine noncld_obs_chk(nbands,pxldat,pxsza,testbits)

      real*4 pxldat(nbands)
      byte testbits(2)
      logical fire,smoke

c     Routine which checks for the possible presence of smoke.

c     Initializations.
      fire = .false.
      smoke = .false.
      ref21 = pxldat(7)
      ref66 = pxldat(2)
      t375 = pxldat(8)
      t11 = pxldat(11)
      tdif = t375 - t11
      a = 7.0 + (ref21 / 2.0)

c     First, check for fires (hot spots).
      if(t375 .gt. 350.0 .and. tdif .gt. 10.0) then
          fire = .true.
      end if

c     Test for thick smoke or fire. If found, clear bit #8.
      if(ref21 .lt. 20.0) then
        if((ref66 .gt. a) .or. fire) then
          smoke = .true.
          call clear_bit(testbits,7)
        end if
      end if

      return
      end
```

```
      subroutine proc_path(water,land,coast,desert,day,snglnt,
     *                     snow,ice,testbits)

c F77 ************************************************************
c ... routine for determining processing path through
c ... the algorithm and setting the appropriate bit
c ... flags. This is where the appropriate day/night
c ... and land/water bits are set.
c
c Input parameters:
c water         Logical variable - true if water background
c land          Logical variable - true if land background
c coast         Logical variable - true if coastal background
c desert        Logical variable - true if desert background
c day           Logical variable - true if sza < 80
c snglnt        Logical variable - true if in sun-glint region
c snow          Logical variable - true if snow background
c ice           Logical variable - true if sea-ice background
c Output Parameters:
c testbits      two word 1-byte array containing bit results
c
c END**************************************************************
c
c ... scalar arguments ..
      logical water,land,coast,desert,day,snglnt,snow,ice

c ... array arguments ..
      byte testbits(2)

c     Set snow/ice bit.
      if((.not. snow) .and. (.not. ice)) then
         call set_bit(testbits,4)
      end if

      if(.not. snglnt) then
c        Not in geometric sun-glint region. Set bit.
         call set_bit(testbits,3)
      end if

      if(day) then
c        Daytime.  Set bit.
         call set_bit(testbits,2)
      end if

c     Set coast, desert, or land processing path flags.  Default is
c     water (00), which is set during initialization of the bit flags.

      if(coast) then
c       Set coast bit.
        call set_bit(testbits,5)
      else if(desert) then
c       Set desert bit.
        call set_bit(testbits,6)
      else if(land) then
c        Set "land" bits.
         call set_bit(testbits,5)
         call set_bit(testbits,6)
      end if

      return
      end
```

```
      subroutine get_stats(confdnc,npix,n1sm,n2sm,n3sm,n4sm)

c     Routine for counting frequencies of various confidence
c     categories.

      data nmpix /0/ , n1s /0/ , n2s /0/ , n3s /0/ , n4s /0/

      nmpix = nmpix + 1
      if(confdnc .gt. 0.99) then
        n1s = n1s + 1
      else if(confdnc .gt. 0.95) then
        n2s = n2s + 1
      else if(confdnc .gt. 0.66) then
        n3s = n3s + 1
      else if(confdnc .lt. 0.01) then
        n4s = n4s + 1
      end if

      npix = nmpix
      n1sm = n1s
      n2sm = n2s
      n3sm = n3s
      n4sm = n4s

      return
      end
```

```
      subroutine set_confdnc(confdnc,testbits)

c     Routine for setting output "bit" flags according
c     to final confidence of clear sky.

      byte testbits(2)

      if(confdnc .gt. 0.99) then
        call set_bit(testbits,0)
        call set_bit(testbits,1)
      else if(confdnc .gt. 0.95) then
        call set_bit(testbits,1)
      else if(confdnc .gt. 0.66) then
        call set_bit(testbits,0)
      end if

      return
      end
```

```
      subroutine write_bits(outf_unit,bitarray,iyrday,ibhms,islhms1,
     *                      outrec,end_file)

C!F77 ************************************************************
c ... routine for writing results of the cloud mask processing
c ... to a binary file.  the file is direct access with a 1432
c ... byte record length (16 bits or 2 1-byte words for each of 716
c ... MAS pixels in a scan line).  in the case of partial proccessing
c ... of scan lines, the file is padded with zeros
c ... The first line (record) contains nothing
c ... but 4 4-byte integers representing the beginning date (yyddd),
c ... time (hms), time in milliseconds and total number of records
c ... written to the file.
c ...
c!Input parameters:
c outf_unit          FORTRAN unit # attached to output file
c bitarray           array containing one scanline of bit results
c iyrday             Beginning date of data (yyddd)
c ibhms              Beginning time of data (hhmmss)
c islhms1            Beginning time of cloud mask (hhmmss)
c outrec             Output record number
c end_fil            End of processing flag - set equal to
c                    .true. when finished
c!Output Parameters:
c ** none ** writes output to file
c
c!END**************************************************************

c ... parameters ..
      integer npixel
      integer nhedr
      integer outf_unit
      parameter (npixel=716)
      parameter (nhedr=358)
c ...
c ... scalar arguments ..
      integer iyrday,ibhms,islhms1,outrec
      logical end_file
c ...
c ... array arguments ..
      byte bitarray(2,npixel)
c ...
c ... local arrays ..
      integer hedr(nhedr)

      save hedr

c ... On first call, write header record containing orbit start
c ... date and time, and cloud mask beginning time in milliseconds.

      if (outrec.eq.2) then

        hedr(1) = iyrday
        hedr(2) = ibhms
        hedr(3) = islhms1

        write (outf_unit,rec=1) hedr
        write (outf_unit,rec=outrec) bitarray

      else if (.not. end_file) then

        write (outf_unit,rec=outrec) bitarray

      end if

      if (end_file) then

c ... The end of data has been reached.  Write total number
c ... of records in the output file to the header.
        hedr(4) = outrec
        write (outf_unit,rec=1) hedr

      end if

      return
      end
```

```
      subroutine stats_out(nmpix,n1s,n2s,n3s,n4s)

c     Routine for calculating and writing output statistics.

      if(nmpix .gt. 0) then
         pcn1s = (float(n1s) / nmpix) * 100.0
         pcn2s = (float(n2s) / nmpix) * 100.0
         pcn3s = (float(n3s) / nmpix) * 100.0
         pcn4s = (float(n4s) / nmpix) * 100.0
      else
         pcn1s = 32767.0
         pcn2s = 32767.0
         pcn3s = 32767.0
         pcn4s = 32767.0
      end if

      write(*,'(1x,''stats: '')')

      write(*,'(1x,''# total pixels: '',i10)') nmpix
      write(*,'(1x,''# pixels > 99%: '',i10)') n1s
      write(*,'(1x,''# pixels > 95%: '',i10)') n2s
      write(*,'(1x,''# pixels > 66%: '',i10)') n3s
      write(*,'(1x,''# pixels <  1%: '',i10)') n4s
      write(*,'(1x)')

      write(*,'(1x,''% with confidence > 99% '',f10.5)') pcn1s
      write(*,'(1x,''% with confidence > 95% '',f10.5)') pcn2s
      write(*,'(1x,''% with confidence > 66% '',f10.5)') pcn3s
      write(*,'(1x,''% with confidence <  1% '',f10.5)') pcn4s

      return
      end
```

```
      subroutine file_close(if_hdfid,tp_unit,eco_unit,
     *                      outf_unit)

c     Routine for closing all files.

      integer*4 tp_unit,eco_unit,outf_unit,if_hdfid,sfend

      close (tp_unit)
      close (outf_unit)
      close (eco_unit)
      irt = sfend(if_hdfid)

      return
      end
```

```c
/*  GIHLS211 transforms latitude, longitude coordinates to line, sample
    Renamed GIHLS211 to GETCOORD !!
    coordinates for an image in the Goode's Interrupted Homolosine
    projection.  This routine was complied and run using the C compiler on
    SunOS 4.2 (UNIX).  Results were accurate at the time of testing, but they
    are by no means guaranteed!

    By D. Steinwand, HSTX/EROS Data Center    June, 1993

References

1.  Snyder, John P. and Voxland, Philip M., "An Album of Map Projections",
    U.S. Geological Survey Professional Paper 1453 , United State Government
    Printing Office, Washington D.C., 1989.

2.  Snyder, John P., "Map Projections--A Working Manual", U.S. Geological
    Survey Professional Paper 1395 (Supersedes USGS Bulletin 1532), United
    State Government Printing Office, Washington D.C., 1987.

3.  Goode, J.P., 1925,  The Homolosine projection:  a new device for
    portraying the Earth's surface entire:  Assoc. Am. Geographers, Annals,
    v. 15, p. 119-125

4.  Steinwand, Daniel R., "Mapping Raster Imagery to the Interrupted
    Goode Homolosine Projection", 1993.  In press--IJRS.
----------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>

#define PI      3.141592653589793238
#define HALF_PI PI*0.5
#define TWO_PI  PI*2.0
#define EPSLN   1.0e-10
#define R2D     57.2957795131
#define D2R     0.0174532925199

#define OK      1
#define ERROR  -1
#define IN_BREAK -2

/* Variables common to all subroutines in this code file
  -----------------------------------------------------*/
static double R;                /* Radius of the earth (sphere) */
static double lon_center[12];   /* Central meridians, one for each region */
static double feast[12];        /* False easting, one for each region */

/* Transformation routine
  -------------------------------*/
void getcoord (double *lt, double *ln, double *lne, double *smp)
{
float pixsiz;
double x,y;
double lat, lon, line, samp;
int nl,ns;
double ul_x, ul_y;
int count;

pixsiz = 1000.0;

/* Report parameters and image geometric characteristics to the user
  -------------------------------------------------------------------*/
/*printf("Converting latitude, longitude to line, sample coordinates\n\n");
printf("Pixel size is %f km\n\n", pixsiz/1000.0);  */
if (pixsiz == 1000.0) {
   ul_x = -20015000.0;
   ul_y =   8673000.0;
   nl = 17347;
   ns = 40031;
   }
else {
   ul_x = -20011500.0;
   ul_y =   8669500.0;
   nl = 2168;
   ns = 5004;
   }
/*printf("Image size is %d lines by %d samples with an upper left\n",nl,ns);
printf("corner of UL_X = %lf and UL_Y = %lf meters.\n", ul_x, ul_y);   */

/* Initialize the Interrupted Goode Homolosine projection
  -------------------------------------------------------*/
goode_init(6370997.0);

/* Process point
  ------------------*/
   lat = *lt;
   lon = *ln;
/* printf("%12.6f %12.6f\n",lat, lon);   */
   lat *= D2R;
   lon *= D2R;
   goode_forward(lon, lat, &x, &y);
   line = (ul_y - y) / pixsiz + 1.0;
   samp = (x - ul_x) / pixsiz + 1.0;
/* printf("%12.6f %12.6f\n",lat, lon);
   printf("%12.6f %12.6f\n",line, samp);    */

   *lne = line;
   *smp = samp;
   *lt = lat;
   *ln = lon;

   return;
}

/* Function to report bad parameters
  ---------------------------------*/
bad_input_parms() {
printf("Syntax: gihll2ls pixsize\n");
printf("         pixsize in km = 1 or 8\n\n");
}

/* Initialize the Goode's Homolosine projection
  ---------------------------------------------*/
goode_init(r)
double r;                       /* (I) Radius of the earth (sphere) */
{
/* Place parameters in static storage for common use
  -------------------------------------------------*/
R = r;

/* Initialize central meridians for each of the 12 regions
  -------------------------------------------------------*/
lon_center[0] = -1.74532925199;       /* -100.0 degrees */
lon_center[1] = -1.74532925199;       /* -100.0 degrees */
lon_center[2] =  0.523598775598;      /*   30.0 degrees */
lon_center[3] =  0.523598775598;      /*   30.0 degrees */
lon_center[4] = -2.79252680319;       /* -160.0 degrees */
lon_center[5] = -1.0471975512;        /*  -60.0 degrees */
lon_center[6] = -2.79252680319;       /* -160.0 degrees */
lon_center[7] = -1.0471975512;        /*  -60.0 degrees */
lon_center[8] =  0.349065850399;      /*   20.0 degrees */
```

```c
lon_center[9] =  2.44346095279;          /*  140.0 degrees */
lon_center[10] = 0.349065850399;         /*   20.0 degrees */
lon_center[11] = 2.44346095279;          /*  140.0 degrees */

/* Initialize false eastings for each of the 12 regions
   -------------------------------------------------*/
feast[0] = R * -1.74532925199;
feast[1] = R * -1.74532925199;
feast[2] = R * 0.523598775598;
feast[3] = R * 0.523598775598;
feast[4] = R * -2.79252680319;
feast[5] = R * -1.0471975512;
feast[6] = R * -2.79252680319;
feast[7] = R * -1.0471975512;
feast[8] = R * 0.349065850399;
feast[9] = R * 2.44346095279;
feast[10] = R * 0.349065850399;
feast[11] = R * 2.44346095279;

/* Report parameters to the user
   -----------------------------*/
/*ptitle("Goode's Homolosine Equal Area");
radius(r):   */
return(OK);
}
```

```c
/* Goode's Homolosine forward equations--mapping lat,long to x,y
   ----------------------------------------------------------*/
goode_forward(lon, lat, x, y)
double lon;                        /* (I) Longitude */
double lat;                        /* (I) Latitude */
double *x;                         /* (O) X projection coordinate */
double *y;                         /* (O) Y projection coordinate */
{
double adjust_lon();     /* Function to adjust longitude to -180 - 180 */
double delta_lon;        /* Delta longitude (Given longitude - center */
double theta;
double delta_theta;
double constant;
int i;
int region;

/* Forward equations
   -----------------*/
if (lat >= 0.710987989993)                          /* if on or above 40 44' 11.8" */
   {
   if (lon <= -0.698131700798) region = 0;    /* If to the left of -40 */
   else region = 2;
   }
else if (lat >= 0.0)                                 /* Between 0.0 and 40 44' 11.8" */
   {
   if (lon <= -0.698131700798) region = 1;    /* If to the left of -40 */
   else region = 3;
   }
else if (lat >= -0.710987989993)                     /* Between 0.0 & -40 44' 11.8" */
   {
   if (lon <= -1.74532925199) region = 4;          /* If between -180 and -100 */
   else if (lon <= -0.349065850399) region = 5;    /* If between -100 and -20 */
   else if (lon <= 1.3962634016) region = 8;       /* If between -20 and 80 */
   else region = 9;                                 /* If between 80 and 180 */
   }
else                                                  /* Below -40 44' */
   {
   if (lon <= -1.74532925199) region = 6;          /* If between -180 and -100 */
   else if (lon <= -0.349065850399) region = 7;       /* If between -100 and -20 */
   else if (lon <= 1.3962634016) region = 10;      /* If between -20 and 80 */
   else region = 11;                                /* If between 80 and 180 */
   }

if (region==1||region==3||region==4||region==5||region==8||region==9)
   {
   delta_lon = adjust_lon(lon - lon_center[region]);
   *x = feast[region] + R * delta_lon * cos(lat);
   *y = R * lat;
   }
else
   {
   delta_lon = adjust_lon(lon - lon_center[region]);
   theta = lat;
   constant = PI * sin(lat);

/* Iterate using the Newton-Raphson method to find theta
   -----------------------------------------------------*/
   for (i=0;;i++)
      {
      delta_theta = -(theta + sin(theta) - constant) / (1.0 + cos(theta));
      theta += delta_theta;
      if (fabs(delta_theta) < EPSLN) break;
      if (i >= 30)
         {giherror("Iteration failed to converge","Goode-forward");return(ERROR);}
```

```
      }
   theta /= 2.0;
   *x = feast[region] + 0.900316316158 * R * delta_lon * cos(theta);
   *y = R * (1.4142135623731 * sin(theta) - 0.0528035274542 * sign(lat));
   }

return(OK);
}

/* Functions to report projection parameters
  -----------------------------------------*/
ptitle(A) char *A; { printf("\n%s Projection Parameters:\n\n",A); }
radius(A) double A;{printf("   Radius of Sphere:     %lf meters\n",A); }

/* Function to report errors
  -------------------------*/
giherror(what, where) char *what,*where; {printf("[%s] %s\n",where,what);}

/* Function to calculate the sine and cosine in one call.  Some computer
   systems have implemented this function, resulting in a faster implementation
   than calling each function separately.  It is provided here for those
   computer systems which don't implement this function
  ----------------------------------------------------*/
#ifndef sun
sincos(val, sin_val, cos_val) double val; double *sin_val; double *cos_val;
{ *sin_val = sin(val); *cos_val = cos(val); return; }
#endif

/* Function to return the sign of an argument
  ------------------------------------------*/
sign(x) double x; { if (x < 0.0) return(-1); else return(1);}

/* Function to adjust longitude to -180 to 180
  -------------------------------------------*/
double adjust_lon(x) double x;{x=(fabs(x)<PI)?x:(x-(sign(x)*TWO_PI));return(x);}

/* Functions to compute constants e0,e1,e2, and M
  ----------------------------------------------*/
double e0fn(x) double x;{return(1.0-0.25*x*(1.0+x/16.0*(3.0+1.25*x)));}
double e1fn(x) double x;{return(0.375*x*(1.0+0.25*x*(1.0+0.46875*x)));}
double e2fn(x) double x;{return(0.05859375*x*x*(1.0+0.75*x));}
double mlfn(e0,e1,e2,phi) double e0,e1,e2,phi;{
                        return(e0*phi-e1*sin(2.0*phi)+e2*sin(4.0*phi));}
```

```c
/*  GIHLS211 transforms latitude, longitude coordinates to line, sample
    Renamed GIHLS211 to GETCOORD !!
    coordinates for an image in the Goode's Interrupted Homolosine
    projection.  This routine was complied and run using the C compiler on
    SunOS 4.2 (UNIX).  Results were accurate at the time of testing, but they
    are by no means guaranteed!

    By D. Steinwand, HSTX/EROS Data Center    June, 1993

References

1.  Snyder, John P. and Voxland, Philip M., "An Album of Map Projections",
    U.S. Geological Survey Professional Paper 1453 , United State Government
    Printing Office, Washington D.C., 1989.

2.  Snyder, John P., "Map Projections--A Working Manual", U.S. Geological
    Survey Professional Paper 1395 (Supersedes USGS Bulletin 1532), United
    State Government Printing Office, Washington D.C., 1987.

3.  Goode, J.P., 1925,  The Homolosine projection:  a new device for
    portraying the Earth's surface entire:  Assoc. Am. Geographers, Annals,
    v. 15, p. 119-125

4.  Steinwand, Daniel R., "Mapping Raster Imagery to the Interrupted
    Goode Homolosine Projection", 1993.  In press--IJRS.
-----------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>

#define PI       3.141592653589793238
#define HALF_PI  PI*0.5
#define TWO_PI   PI*2.0
#define EPSLN    1.0e-10
#define R2D      57.2957795131
#define D2R      0.0174532925199

#define OK       1
#define ERROR   -1
#define IN_BREAK -2

/* Variables common to all subroutines in this code file
  -----------------------------------------------------*/
static double R;                /* Radius of the earth (sphere) */
static double lon_center[12];   /* Central meridians, one for each region */
static double feast[12];        /* False easting, one for each region */

/* Transformation routine
  -------------------------------*/
void nacoord_(double *lt, double *ln, double *lne, double *smp)
{
float pixsiz;
double x,y;
double lat, lon, line, samp;
int nl,ns;
double ul_x, ul_y;
int count;

pixsiz = 1000.0;

/* Report parameters and image geometric characteristics to the user
  -----------------------------------------------------------------*/
/*printf("Converting latitude, longitude to line, sample coordinates\n\n");
printf("Pixel size is %f km\n\n", pixsiz/1000.0);  */
if (pixsiz == 1000.0) {
   ul_x = -17359000.0;
   ul_y = 8423000.0;
   nl = 7793;
   ns = 11329;
   }
else {
   ul_x = -20011500.0;
   ul_y =   8669500.0;
   nl = 2168;
   ns = 5004;
   }
/*printf("Image size is %d lines by %d samples with an upper left\n",nl,ns);
printf("corner of UL_X = %lf and UL_Y = %lf meters.\n", ul_x, ul_y);    */

/* Initialize the Interrupted Goode Homolosine projection
  -----------------------------------------------------*/
ngoode_init(6370997.0);

/* Process point
  -----------------*/
   lat = *lt;
   lon = *ln;
/* printf("%12.6f %12.6f\n",lat, lon);    */
   lat *= D2R;
   lon *= D2R;
   ngoode_forward(lon, lat, &x, &y);
   line = (ul_y - y) / pixsiz + 1.0;
   samp = (x - ul_x) / pixsiz + 1.0;
/* printf("%12.6f %12.6f\n",lat, lon);
   printf("%12.6f %12.6f\n",line, samp);    */

   *lne = line;
   *smp = samp;
   *lt = lat;
   *ln = lon;

   return;
}

/* Function to report bad parameters
  -------------------------------*/
nbad_input_parms() {
printf("Syntax: gihll2ls pixsize\n");
printf("         pixsize in km = 1 or 8\n\n");
}

/* Initialize the Goode's Homolosine projection
  -------------------------------------------*/
ngoode_init(r)
double r;                       /* (I) Radius of the earth (sphere) */
{
/* Place parameters in static storage for common use
  -------------------------------------------------*/
R = r;

/* Initialize central meridians for each of the 12 regions
  -------------------------------------------------------*/
lon_center[0] = -1.74532925199;        /* -100.0 degrees */
lon_center[1] = -1.74532925199;        /* -100.0 degrees */
lon_center[2] =  0.523598775598;       /*   30.0 degrees */
lon_center[3] =  0.523598775598;       /*   30.0 degrees */
lon_center[4] = -2.79252680319;        /* -160.0 degrees */
lon_center[5] = -1.0471975512;         /*  -60.0 degrees */
lon_center[6] = -2.79252680319;        /* -160.0 degrees */
lon_center[7] = -1.0471975512;         /*  -60.0 degrees */
lon_center[8] =  0.349065850399;       /*   20.0 degrees */
```

```
lon_center[9] =  2.44346095279;          /*  140.0 degrees */
lon_center[10] = 0.349065850399;         /*   20.0 degrees */
lon_center[11] = 2.44346095279;          /*  140.0 degrees */

/* Initialize false eastings for each of the 12 regions
  ------------------------------------------------*/
feast[0] = R * -1.74532925199;
feast[1] = R * -1.74532925199;
feast[2] = R * 0.523598775598;
feast[3] = R * 0.523598775598;
feast[4] = R * -2.79252680319;
feast[5] = R * -1.0471975512;
feast[6] = R * -2.79252680319;
feast[7] = R * -1.0471975512;
feast[8] = R * 0.349065850399;
feast[9] = R * 2.44346095279;
feast[10] = R * 0.349065850399;
feast[11] = R * 2.44346095279;

/* Report parameters to the user
  ----------------------------*/
/*nptitle("Goode's Homolosine Equal Area");
nradius(r):   */
return(OK);
}
```

```
/* Goode's Homolosine forward equations--mapping lat,long to x,y
  ----------------------------------------------------------*/
ngoode_forward(lon, lat, x, y)
double lon;                        /* (I) Longitude */
double lat;                        /* (I) Latitude */
double *x;                         /* (O) X projection coordinate */
double *y;                         /* (O) Y projection coordinate */
{
double nadjust_lon();   /* Function to adjust longitude to -180 - 180 */
double delta_lon;       /* Delta longitude (Given longitude - center */
double theta;
double delta_theta;
double constant;
int i;
int region;

/* Forward equations
  -----------------*/
if (lat >= 0.710987989993)                     /* if on or above 40 44' 11.8" */
   {
   if (lon <= -0.698131700798) region = 0;     /* If to the left of -40 */
   else region = 2;
   }
else if (lat >= 0.0)                           /* Between 0.0 and 40 44' 11.8" */
   {
   if (lon <= -0.698131700798) region = 1;     /* If to the left of -40 */
   else region = 3;
   }
else if (lat >= -0.710987989993)               /* Between 0.0 & -40 44' 11.8" */
   {
   if (lon <= -1.74532925199) region = 4;        /* If between -180 and -100 */
   else if (lon <= -0.349065850399) region = 5;  /* If between -100 and -20 */
   else if (lon <= 1.3962634016) region = 8;     /* If between -20 and 80 */
   else region = 9;                              /* If between 80 and 180 */
   }
else                                             /* Below -40 44' */
   {
   if (lon <= -1.74532925199) region = 6;        /* If between -180 and -100 */
   else if (lon <= -0.349065850399) region = 7;      /* If between -100 and -20 */
   else if (lon <= 1.3962634016) region = 10;    /* If between -20 and 80 */
   else region = 11;                             /* If between 80 and 180 */
   }

if (region==1||region==3||region==4||region==5||region==8||region==9)
   {
   delta_lon = nadjust_lon(lon - lon_center[region]);
   *x = feast[region] + R * delta_lon * cos(lat);
   *y = R * lat;
   }
else
   {
   delta_lon = nadjust_lon(lon - lon_center[region]);
   theta = lat;
   constant = PI * sin(lat);

/* Iterate using the Newton-Raphson method to find theta
  -----------------------------------------------------*/
   for (i=0;;i++)
      {
      delta_theta = -(theta + sin(theta) - constant) / (1.0 + cos(theta));
      theta += delta_theta;
      if (fabs(delta_theta) < EPSLN) break;
      if (i >= 30)
         {ngiherror("Iteration failed to converge","Goode-forward");return(ERROR);}
```

```
      }
   theta /= 2.0;
   *x = feast[region] + 0.900316316158 * R * delta_lon * cos(theta);
   *y = R * (1.4142135623731 * sin(theta) - 0.0528035274542 * nsign(lat));
   }

return(OK);
}

/* Functions to report projection parameters
  -------------------------------------------*/
nptitle(A) char *A; { printf("\n%s Projection Parameters:\n\n",A); }
nradius(A) double A;{printf("   Radius of Sphere:     %lf meters\n",A); }

/* Function to report errors
  --------------------------*/
ngiherror(what, where) char *what,*where; {printf("[%s] %s\n",where,what);}

/* Function to calculate the sine and cosine in one call.  Some computer
   systems have implemented this function, resulting in a faster implementation
   than calling each function separately.  It is provided here for those
   computer systems which don't implement this function
  ----------------------------------------------------*/
#ifndef sun
nsincos(val, sin_val, cos_val) double val; double *sin_val; double *cos_val;
{ *sin_val = sin(val); *cos_val = cos(val); return; }
#endif

/* Function to return the sign of an argument
  ------------------------------------------*/
nsign(x) double x; { if (x < 0.0) return(-1); else return(1);}

/* Function to adjust longitude to -180 to 180
  -------------------------------------------*/
double nadjust_lon(x) double x;{x=(fabs(x)<PI)?x:(x-(nsign(x)*TWO_PI));return(x);}

/* Functions to compute constants e0,e1,e2, and M
  -----------------------------------------------*/
double ne0fn(x) double x;{return(1.0-0.25*x*(1.0+x/16.0*(3.0+1.25*x)));}
double ne1fn(x) double x;{return(0.375*x*(1.0+0.25*x*(1.0+0.46875*x)));}
double ne2fn(x) double x;{return(0.05859375*x*x*(1.0+0.75*x));}
double nmlfn(e0,e1,e2,phi) double e0,e1,e2,phi;{
                        return(e0*phi-e1*sin(2.0*phi)+e2*sin(4.0*phi));}
```

```fortran
      real function bright50(band,cwl,r)

c     r         radiance in Watts per square meter per steradian per wavenumber
c     band      MAS band number (26-50)
c     bright50  temperature in Kelvin

      parameter(h = 6.6260755e-34)
      parameter(c = 2.99792458e+8)
      parameter(rk = 1.380658e-23)
      parameter(c1 = 2.0d0 * h * c * c)
      parameter(c2 = h * c / rk)

      integer*4 band
      real*4 ti(25),ts(25)

      data ti /
     *   7.876993e-01,  8.203697e-01,  6.643460e-01,  5.973398e-01,
     *   4.859185e-01,  4.063021e-01,  3.754345e-01,  3.043327e-01,
     *   2.783363e-01,  2.122330e-01,  2.006351e-01,  1.820766e-01,
     *   1.528009e-01,  1.314414e-01,  1.208291e-01,  9.452466e-02,
     *   2.024190e-02, -4.009836e-02, -5.146342e-02, -6.196085e-02,
     *  -6.241275e-02, -5.677923e-02, -6.189410e-02, -7.487083e-02,
     *  -5.352456e-02/

      data ts /
     *   9.992109e-01,  9.991444e-01,  9.992756e-01,  9.993234e-01,
     *   9.994270e-01,  9.995031e-01,  9.995242e-01,  9.996009e-01,
     *   9.996225e-01,  9.997025e-01,  9.997097e-01,  9.997280e-01,
     *   9.997643e-01,  9.997908e-01,  9.998015e-01,  9.998395e-01,
     *   9.997897e-01,  9.998696e-01,  9.999445e-01,  9.999673e-01,
     *   1.000004e+00,  1.000021e+00,  1.000030e+00,  1.000044e+00,
     *   1.000037e+00/

      if(band .lt. 26 .or. band .gt. 50) then
        write(*,'(1x,''MAS IR band must be in the range 26 - 50'')')
        go to 100
      end if

      ws = 1.0e-06 * cwl
      rs = 1.0e06 * r
      tc = c2 / ( ws * alog( c1 / ( rs * ws**5 ) + 1.0e+0 ) )
      bright50 = ( tc - ti( band - 25 ) ) / ts( band - 25 )
c     if(band .eq. 49) then
c       write(*,'(1x,''bright50 '',i10,4f8.3)') band,cwl,r,tc,bright50
c     end if

 100  return
      end
```

```
      subroutine snowb(nbands,pxldat,rlat,sza,jday,avgtherm,
     *                 dh_snow)
c
c F77 ************************************************************
c
c Bryan Baum's snow test.

c Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c rlat          latitude
c sza           solar zenith angle
c avgtherm      Array containing regional means of IR brightness temps.
c jday          Day of year
c Output Parameters:
c dh_snow       logical variable indicating the prescence of snow or
c               ice in a FOV.
c
c END***************************************************************
c
      include 'thresholds.inc'
c
c ... parameters ..
c ...
c ... scalar arguments ..
      logical dh_snow
c ...
c ... array arguments ..
      integer jday
      real pxldat(nbands),avgtherm(nbands),rlat,sza
c ...
c ... local scalars ..
      real masir11,masv66,masir37,ref3,ref3o1,irdif
c
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masir37 = avgtherm(8)
      masir11 = avgtherm(11)
      call get_refl3(masir37,masir11,jday,sza,ref3)
      ref3o1 = ref3 / masv66
      irdif = masir37 - masir11
c
      if(abs(rlat) .gt. 50.0 .and. masir11 .le. 260.0) then

        if(ref3o1 .le. .06 .or. (masv66 .ge. 25.0 .and.
     *     ref3 .le. 5.0 .and. irdif .le. 16.0)) then

          dh_snow = .true.

        else

          dh_snow = .false.

        end if

      else

        if(masv66 .ge. 20.0 .and. masir11 .le. 277.0 .and.
     *     ref3 .le. 3.0 .and. irdif .le. 8.0) then

          dh_snow = .true.

        else

          dh_snow = .false.

        end if

      end if

      return
      end
```

```
      subroutine chk_cnstnc(nlcntx,necntx,rgvza,rgsza,rgrfa,rgpcwt,
     *                      rgeco,rglat,rglon,cnpcwt,cnlwc)

c     Routine which checks consistency of various ancillary data and
c     sets flags accordingly.

      integer*4 rgpcwt(nlcntx,necntx),pxpcwt
      real*4 rgvza(nlcntx,necntx),
     *       rgrfa(nlcntx,necntx),rgsza(nlcntx,necntx),
     *       rglat(nlcntx,necntx),rglon(nlcntx,necntx)
      logical*4 cnpcwt,cnlwc
      byte rgeco(nlcntx,necntx),eco_type

c     Initializations.
      nwater = 0
      ncoast = 0
      nland = 0
      itotal = nlcntx * necntx

c     Loop over each pixel position in the current context.

      do i = 1,nlcntx
        do j = 1,necntx

c         Check land/sea tag (% coverage of water in pixel).
          pxpcwt = rgpcwt(i,j)
          if(pxpcwt .eq. 100) then
            nwater = nwater + 1
          else if(pxpcwt .eq. 50) then
            ncoast = ncoast + 1
          else if(pxpcwt .eq. 0) then
            nland = nland + 1
          end if

        enddo
      enddo

      if( (nwater+ncoast) .eq. itotal) then
        cnlwc = .true.
      else if( (nland+ncoast) .eq. itotal) then
        cnlwc = .true.
      else
        cnlwc = .false.
      end if

      if(nwater .eq. itotal) then
        cnpcwt = .true.
      else
        cnpcwt = .false.
      end if

c     write(*,'(1x,''consistency '',2l10)') cnlwc,cnpcwt

      return
      end
```

```
      subroutine PolarNite_snow(nbands,pxldat,vza,eco_type,
     +                          avgtherm,testbits,confdnc)

C F77 **********************************************************
c
c       Routine for performing clear sky tests over snow
c       surfaces during nighttime hours.
c
c
c Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c vza           Viewing zenith angle
c eco_type      Holder of ecosystem type (1-17)
c avgtherm      Average brightness temperature for given box
c!Output Parameters:
c testbits      two word 1-byte array containing bit results
c confdnc       product of all applied individual confidences
c
c END**************************************************************
c
      include 'thresholds.inc'
c ...
c ... scalar arguments ..
      real confdnc,vza
      integer nbands
      byte eco_type
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,dtr,mas4_12,masir11,masir12,masir13,masir4,
     +     pi,c3,mas11_4,pre_confdnc,fac,groups
      integer nptests
c ...
c ... local arrays ..
      integer ngtests(5)

c ... external subroutines ..
      external conf_test,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group.
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0
      ngtests(5) = 0

c ... Set confidence to 1.0.
      confdnc = 1.0

c ... Place band values into individual variables for easy
c ... identification.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masir4 = pxldat(9)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... The "cmin" variables represent group test confidences.
      cmin1 = 1.0
      cmin2 = 1.0
      cmin5 = 1.0

C     **** GROUP 1 TESTS ************************************

      nptests = 0

c ... Co2 high cloud test.
      if (masir13.gt.pnsco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,pnsco2(1),pnsco2(3),pnsco2(4),
     *               pnsco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  **************************
c
c
c     ****  GROUP 2 TESTS  **********************************

      nptests = 0

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      mas11_4 = masir11 - masir4
      if (mas11_4.le.pns11_4lo(2)) then
        nptests = nptests + 1
      end if
      call conf_test(mas11_4,pns11_4lo(1),pns11_4lo(3),pns11_4lo(4),
     +               pns11_4lo(2),1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******    END OF GROUP 2 TESTS ***************************
c
c
c *******   START OF GROUP 5 TESTS  ************************

      nptests = 0

c ... 4-12um brightness temperature difference test
c ... for thin cirrus.
      mas4_12 = masir4 - masir12
      if (mas4_12.le.pns4_12hi(2)) then
        nptests = nptests + 1
      end if
      call conf_test(mas4_12,pns4_12hi(1),pns4_12hi(3),pns4_12hi(4),
     +               pns4_12hi(2),1,c3)
      cmin5 = min(cmin5,c3)
      ngtests(5) = ngtests(5) + 1
```

```
      if(nptests .eq. ngtests(5) .and. ngtests(5) .ne. 0) then
        call set_bit(testbits,14)
      end if

c ********    END OF GROUP 5 TESTS  *****************************

c     Determine final confidence based on group values.
      pre_confdnc = cmin1 * cmin2 * cmin5
      groups = 0.0
      do kk = 1,5
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

      return
      end
```

```fortran
      subroutine LandNite_type1(nbands,pxldat,avgtherm,vza,eco_type,
     +                          testbits,confdnc)
C!F77 ***********************************************************
c      Routine for performing clear sky tests over land
c      surfaces during nighttime hours.
c
c      A new algorithm is being tested where groups of tests
c      will be used.  The groupings represent spectral tests which
c      attempt to detect the same type of clouds.  The minimum
c      confidence result from all grouped tests will be used to
c      determine a group confidence which will then be multiplied
c      with other group results to determine a final pixel clear
c      confidence value.
c
c      For nighttime land the groups are:
c          Group 1: High thick cloud
c                   13.9 micron bt test (masir13)
c
c          Group 2: Low cloud - thick
c                   8-11 micron and 11-12 micron bt tests
c                   11-4 micron bt tests
c
c          Group 5: High cloud - thin
c                   3.7-12 micron bt test
c
c!Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c vza           Current pixel viewing angle
c eco_type      Ecosystem type (1-17)
c avgtherm      Average brightness temperature over given box
c!Output Parameters:
c testbits      two word 1-byte array containing bit results
c confdnc       product of all applied individual confidences
c
c!END*************************************************************
c
      INCLUDE 'thresholds.inc'

c ...
c ... scalar arguments ..
      real confdnc,vza
      integer nbands
      byte eco_type
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,dtr,mas4_12,masir11,masir12,masir13,masir4,
     +     pi,c3,mas11_4,groups,fac,pre_confdnc
      integer nptests
c ...
c ... local arrays ..
      integer ngtests(5)
c ...
c ... external subroutines ..
      external conf_test,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos
c ...
c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0
      ngtests(5) = 0

c ... confidence to 1.0 to begin with
      confdnc = 1.0

c ... place band values into individual variables for easy
c ... identification
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masir4 = pxldat(9)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... the "cmin" variables represent group test confidences
      cmin1 = 1.0
      cmin2 = 1.0
      cmin5 = 1.0

C     **** GROUP 1 TESTS ************************************

      nptests = 0

c ... co2 high cloud test
      if (masir13.gt.nlco2(2)) then
        nptests = nptests + 1
      end if
c     write(*,'(1x,''masir13 '',4f8.2)') masir13,nlco2(2),
c    *       nlco2(1),nlco2(3)
      call conf_test(masir13,nlco2(1),nlco2(3),nlco2(4),
     *               nlco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  **************************
c
c     ****  GROUP 2 TESTS  **********************************

      nptests = 0

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      mas11_4 = masir11 - masir4
      if (mas11_4.le.nl11_4lo(2)) then
        nptests = nptests + 1
      end if
c     write(*,'(1x,''mas11_4: '',6f10.2)') mas11_4,nl11_4lo(2),
c    *       nl11_4lo(1),nl11_4lo(3),masir11,masir4
      call conf_test(mas11_4,nl11_4lo(1),nl11_4lo(3),nl11_4lo(4),
     +               nl11_4lo(2),1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
```

```
        call set_bit(testbits,11)
      end if

c *******     END OF GROUP 2 TESTS ****************************

c *******    START OF GROUP 5 TESTS  *************************

      nptests = 0

c ... 4-12um brightness temperature difference test
c ... for thin cirrus).
      mas4_12 = masir4 - masir12
      if (mas4_12.le.nl4_12hi(2)) then
        nptests = nptests + 1
      end if
c     write(*,'(1x,''mas4_12: '',6f10.2)') mas4_12,nl4_12hi(2),
c    *       nl4_12hi(1),nl4_12hi(3),masir4,masir12
      call conf_test(mas4_12,nl4_12hi(1),nl4_12hi(3),nl4_12hi(4),
     +              nl4_12hi(2),1,c3)
      cmin5 = min(cmin5,c3)
      ngtests(5) = ngtests(5) + 1

      if(nptests .eq. ngtests(5) .and. ngtests(5) .ne. 0) then
        call set_bit(testbits,14)
      end if

c ********    END OF GROUP 5 TESTS  ****************************

c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin5
      groups = 0.0
      do kk = 1,5
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     write(*,'(1x,''confdnc '',8f8.5)') c1,c2,c3,
c    +       cmin1,cmin2,cmin5,fac,confdnc

      return
      end
```

```
      subroutine ocean_nite(nbands,pxldat,avgtherm,vza,pw,testbits,
     +                      confdnc)
c F77 ***********************************************************
c
c      Routine for performing clear sky tests over water
c      surfaces during nightime hours.
c
c Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c vza           Current pixel viewing angle
c pw            Amount of precipitable water at pixel site
c avgtherm      Average bt for a given box size
c!Output Parameters:
c testbits      two word 1-byte array containing bit results
c confdnc       product of all applied individual confidences
c
c END*************************************************************
c
      INCLUDE 'thresholds.inc'
c
c ... scalar arguments ..
      real confdnc,vza,pw
      real avgtherm(nbands)
      integer nbands
c ...
c ... array arguments ..
      real pxldat(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,ct1,ct2,diftsp1,diftsp2,dtr,mas11_4,masdf1,
     +     masdf2,masir11,masir12,masir13,masir4,pi,c5,pre_confdnc,
     +     groups,fac,masir8
      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2)
      integer ngtests(2)
c ...
c ... external functions ..
      real rega,regb
      external rega,regb
c ...
c ... external subroutines ..
      external conf_test,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0

c ... Place band values into individual variables for easy
c ... identificatio.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masir4 = pxldat(9)
      masir8 = pxldat(10)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... the "cmin" variables represent group test confidences
      cmin1 = 1.0
      cmin2 = 1.0

C     **** GROUP 1 TESTS *************************************

      nptests = 0

c     11 micron brigthness temperature threshold test.
      if (masir11 .ge. nobt11(2)) then
        nptests = nptests + 1
      end if
c ... calculate confidence compared to low and high confidence cutoffs
      call conf_test(masir11,nobt11(1),nobt11(3),nobt11(4),
     *               nobt11(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

c ... co2 high cloud test
      if (masir13.gt.noco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,noco2(1),noco2(3),noco2(4),
     *               noco2(2),1,c2)
      cmin1 = min(cmin1,c2)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  **************************
c
c     ****  GROUP 2 TESTS  **********************************

      nptests = 0

c ... Tri-spectral tests - apply only if a pw value exists for this
c ... pixel.
      if (pw .gt. 1.0) then

c ...   get dynamic clear sky thresholds based on pw
        diftsp2 = rega(pw)
        diftsp1 = regb(pw)
c ...   calculate 8 minus 11 and 11 minus 12 micron BTDIFs
        masdf2 = masir8 - masir11
        masdf1 = masir11 - masir12
        if ((masdf2.lt.diftsp2) .and. (masdf1.gt.diftsp1)) then
          nptests = nptests + 1
        end if
        locut(1) = diftsp2 + .5
        hicut(1) = diftsp2 - .5
        call conf_test(masdf2,locut,hicut,1.0,diftsp2,1,ct1)
        locut(1) = diftsp1 - .5
        hicut(1) = diftsp1 + .5
        call conf_test(masdf1,locut,hicut,1.0,diftsp1,1,ct2)
        c3 = ct1*ct2
        cmin2 = min(cmin2,c3)
        ngtests(2) = ngtests(2) + 1
        c4 = 0.0
```

```
      else

c ...   If no pw value exists for this pixel, use the low cloud
c ...   11-12 micron BTDIF test.

        masdf1 = masir11 - masir12
        if (masdf1.gt.no11_12lo(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masdf1,no11_12lo(1),no11_12lo(3),no11_12lo(4),
     +                 no11_12lo(2),1,c4)
        cmin2 = min(cmin2,c4)
        ngtests(2) = ngtests(2) + 1
        c3 = 0.0

      end if

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      mas11_4 = masir11 - masir4
      if (mas11_4.le.no11_4lo(2)) then
        nptests = nptests + 1
      end if
      call conf_test(mas11_4,no11_4lo(1),no11_4lo(3),no11_4lo(4),
     +               no11_4lo(2),1,c5)
      cmin2 = min(cmin2,c5)
      ngtests(2) = ngtests(2) + 1

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******      END OF GROUP 2 TESTS ****************************
c
c
c     Determine final confidence based on group values.
      pre_confdnc = cmin1 * cmin2
      groups = 0.0
      do kk = 1,2
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

      return
      end
```

```
      subroutine get_regdif(nlcntx,necntx,inband,nbands,
     *                      rgdata,diff)

c     Computes reflectance or brightness temperature differences between
c     the (center) pixel of interest and the surrounding pixel
c     values.

      real*4 diff(nbands,nlcntx*necntx-1),
     *       rgdata(nlcntx,necntx,nbands),a,b

      midi = nlcntx - ((nlcntx-1) / 2)
      midj = necntx - ((necntx-1) / 2)

      do k = 1,inband

        jj = 0
        do i = 1,nlcntx
          do j = 1,necntx

            if(midi .ne. i .or. midj .ne. j) then

              jj = jj + 1
              a = rgdata(i,j,k)
              b = rgdata(midi,midj,k)

              if(a .ne. 32767.0 .and. b .ne. 32767.0) then
                diff(k,jj) = a - b
              else
                diff(k,jj) = 32767.0
              end if

            end if

          enddo
        enddo
      enddo

      return
      end
```

```
      subroutine spatial_var(nbands,nlcntx,necntx,diff,result)

c F77 ***********************************************************
c
c ... routine for performing spatial variability tests and
c ... setting the appropriate flag.
c
c Input parameters:
c nbands        Total number MAS spectral bands
c nlcntx        Number scan lines in region (context).
c necntx        Number pixels in each scan of region (context).
c diff          Array of surrounding pixel reflectance or brightness
c               temperature differences from center pixel value
c Output Parameters:
c result        result  of spatial variability test (1=uniform)
c
c END*************************************************************

      INCLUDE 'thresholds.inc'

c ... get proper threshold value.
      parameter (masir11=11)
c ...
c ... scalar arguments ..
      integer result,nbands,nlcntx,necntx
c ...
c ... array arguments ..
      real diff(nbands,nlcntx*necntx-1)
c ...
c ... local scalars ..
      integer i,ipt
c ...
c ... intrinsic functions ..
      intrinsic btest,ibset

      ipt = 0
      ndif = (nlcntx*necntx) - 1
c ... Compare surrounding bt differences to threshold.
      do 200 i = 1,ndif
        if (diff(masir11,i).le.dovar11(1)) then
          ipt = ipt + 1
        end if
  200 continue

c ... If all surrounding pixel differences were less than the
c ... threshold value, scene is declared to be uniform.
      if (ipt.eq.ndif) then
        result = 1
      else
        result = 0
      end if

      return
      end
```

```
      subroutine ocean_nite(nbands,pxldat,avgtherm,vza,pw,testbits,
     +                     confdnc)
c F77 ***************************************************************
c
c      Routine for performing clear sky tests over water
c      surfaces during nighttime hours.
c
c Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c vza           Current pixel viewing angle
c pw            Amount of precipitable water at pixel site
c avgtherm      Average bt for a given box size
c!Output Parameters:
c testbits      two word 1-byte array containing bit results
c confdnc       product of all applied individual confidences
c
c END*****************************************************************
c
      INCLUDE 'thresholds.inc'
c
c ... scalar arguments ..
      real confdnc,vza,pw
      real avgtherm(nbands)
      integer nbands
c ...
c ... array arguments ..
      real pxldat(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,ct1,ct2,diftsp1,diftsp2,dtr,mas11_4,masdf1,
     +     masdf2,masir11,masir12,masir13,masir4,pi,c5,pre_confdnc,
     +     groups,fac,masir8
      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2)
      integer ngtests(2)
c ...
c ... external functions ..
      real rega,regb
      external rega,regb
c ...
c ... external subroutines ..
      external conf_test,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0

c ... Place band values into individual variables for easy
c ... identificatio.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masir4 = pxldat(9)
      masir8 = pxldat(10)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... the "cmin" variables represent group test confidences
      cmin1 = 1.0
      cmin2 = 1.0

C     **** GROUP 1 TESTS ************************************

      nptests = 0

c     11 micron brigthness temperature threshold test.
      if (masir11 .ge. nobt11(2)) then
        nptests = nptests + 1
      end if
c ... calculate confidence compared to low and high confidence cutoffs
      call conf_test(masir11,nobt11(1),nobt11(3),nobt11(4),
     *               nobt11(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

c ... co2 high cloud test
      if (masir13.gt.noco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,noco2(1),noco2(3),noco2(4),
     *               noco2(2),1,c2)
      cmin1 = min(cmin1,c2)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  **************************
c
c     ****  GROUP 2 TESTS  **********************************

      nptests = 0

c ... Tri-spectral tests - apply only if a pw value exists for this
c ... pixel.
      if (pw .gt. 1.0) then

c ...   get dynamic clear sky thresholds based on pw
        diftsp2 = rega(pw)
        diftsp1 = regb(pw)
c ...   calculate 8 minus 11 and 11 minus 12 micron BTDIFs
        masdf2 = masir8 - masir11
        masdf1 = masir11 - masir12
        if ((masdf2.lt.diftsp2) .and. (masdf1.gt.diftsp1)) then
          nptests = nptests + 1
        end if
        locut(1) = diftsp2 + .5
        hicut(1) = diftsp2 - .5
        call conf_test(masdf2,locut,hicut,1.0,diftsp2,1,ct1)
        locut(1) = diftsp1 - .5
        hicut(1) = diftsp1 + .5
        call conf_test(masdf1,locut,hicut,1.0,diftsp1,1,ct2)
        c3 = ct1*ct2
        cmin2 = min(cmin2,c3)
        ngtests(2) = ngtests(2) + 1
        c4 = 0.0
```

```fortran
      else

c ...   If no pw value exists for this pixel, use the low cloud
c ...   11-12 micron BTDIF test.

        masdf1 = masir11 - masir12
        if (masdf1.gt.no11_12lo(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masdf1,no11_12lo(1),no11_12lo(3),no11_12lo(4),
     +                 no11_12lo(2),1,c4)
        cmin2 = min(cmin2,c4)
        ngtests(2) = ngtests(2) + 1
        c3 = 0.0

      end if

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      mas11_4 = masir11 - masir4
      if (mas11_4.le.no11_4lo(2)) then
        nptests = nptests + 1
      end if
      call conf_test(mas11_4,no11_4lo(1),no11_4lo(3),no11_4lo(4),
     +               no11_4lo(2),1,c5)
      cmin2 = min(cmin2,c5)
      ngtests(2) = ngtests(2) + 1

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******      END OF GROUP 2 TESTS ****************************
c
c
c     Determine final confidence based on group values.
      pre_confdnc = cmin1 * cmin2
      groups = 0.0
      do kk = 1,2
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

      return
      end
```

```
      subroutine PolarDay_snow(nbands,pxldat,vza,visusd,eco_type,
     +                         avgtherm,testbits,confdnc)

c F77 ************************************************************
c
c      Routine for performing clear sky tests over snow or ice
c      surfaces during daylight hours.
c
c Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c vza           Current pixel viewing angle
c visusd        Logical variable indicating whether vis data used or not
c eco_type      Holder of ecosystem type (1-17)
c avgtherm      Average brightness temperature for given box
c!Output Parameters:
c testbits      two word 1-byte array containing bit results
c confdnc       product of all applied individual confidences
c
c END***************************************************************
c
      include 'thresholds.inc'
c ...
c ... scalar arguments ..
      real confdnc,vza
      integer nbands
      logical visusd
      byte eco_type
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,cosvza,dfthrsh,diftemp,dtr,mas11_4,masdf1,
     +     masir11,masir12,masir13,masir4,masir8,masv162,masv188,
     +     masir3,masv95,masv66,masv88,pi,
     +     masv55,masir65,pre_confdnc,fac,groups
      integer nptests
c ...
c ... local arrays ..
      integer ngtests(4)

c ... external subroutines ..
      external conf_test,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group.
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... Place band values into individual variables for easy
c ... identification.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv188 = pxldat(6)
      masv88 = pxldat(3)
      masir4 = pxldat(9)
      masir11 = avgtherm(11)
      masir13 = avgtherm(13)

c ... The "cmin" variables represent test group confidences.
      cmin1 = 1.0
      cmin2 = 1.0
      cmin4 = 1.0

c ... Perform tests.

c     *****  GROUP 1 TESTS  *********************************

      nptests = 0

c ... co2 high cloud test
      if (masir13.gt.pdsco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,pdsco2(1),pdsco2(3),pdsco2(4),
     *               pdsco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  **************************
c
c
c
c     ****  GROUP 2 TESTS  **********************************

      nptests = 0

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      if (visusd) then
        mas11_4 = masir4 - masir11
        if (mas11_4.le.pds4_11(2)) then
          nptests = nptests + 1
        end if
        call conf_test(mas11_4,pds4_11(1),pds4_11(3),pds4_11(4),
     +                 pds4_11(2),1,c2)
        cmin2 = min(cmin2,c2)
        ngtests(2) = ngtests(2) + 1
      end if

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******     END OF GROUP 2 TESTS ***************************
c
c
c
c
c **********    START OF GROUP 4 TESTS   **********************

      nptests = 0

c ... Near-infrared high cloud test.
      if (visusd) then
        if (masv188.le.pdsref3(2)) then
```

```
          nptests = nptests + 1
        end if
        call conf_test(masv188,pdsref3(1),pdsref3(3),pdsref3(4),
     +                 pdsref3(2),1,c3)
        cmin4 = min(cmin4,c3)
        ngtests(4) = ngtests(4) + 1
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ************   END OF GROUP 4 TESTS   ****************************
c
c
c     Determine final confidence based on group values.
      pre_confdnc = cmin1 * cmin2 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv88 .lt. doref2(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```
      subroutine DesertDay_c(nbands,pxldat,vza,visusd,avgtherm,
     +                       testbits,confdnc)

C!F77 ************************************************************
c
c Description:
c      Routine for performing clear sky tests over coastal desert
c      surfaces during daylight hours.
c
c Input parameters:
c nbands        Number of MAS channels
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c avgtherm      Average BT over region
c vza           Current pixel viewing angle
c visusd        Logical variable indicating whether vis data used or not
c
c Output Parameters:
c testbits      two word 1-byte array containing bit results
c confdnc       product of all applied individual confidences
c
c!End---------------------------------------------------------------

      include 'thresholds.inc'

c ... scalar arguments ..
      integer nbands
      real confdnc,vza
      logical visusd
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,c5,cosvza,dfthrsh,diftemp,dtr,mas11_4,masdf1,
     +     masir11,masir12,masir13,masir4,masv188,masv66,masv88,
     +     pi,schi,vrat,c6,cmin1,cmin2,cmin3,cmin4,hiconf,loconf,
     +     fac,pre_confdnc,groups

      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2),midpt(2)
      integer ngtests(4)
c ...
c ... external subroutines ..
      external conf_test,tview,set_bit,clear_bit
c ...
c ... intrinsic functions ..
      intrinsic cos,acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... set confidence to 1.0 to begin with
      confdnc = 1.0

c ... place band values into individual variables for easy
c ... identification
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv188 = pxldat(6)
      masir4 = pxldat(9)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... the "cmin" variables represent group test confidences
      cmin1 = 1.0
      cmin2 = 1.0
      cmin3 = 1.0
      cmin4 = 1.0

c ... perform tests.
c
c     **** GROUP 1 TESTS ****************************************

      nptests = 0

c ... co2 high cloud test
      if (masir13.gt.dsco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,dsco2(1),dsco2(3),dsco2(4),
     *               dsco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  **************************
c
c
c
c     ****  GROUP 2 TESTS  ************************************

      nptests = 0

c ... 11-12um brightness temperature difference test
c ... for thin cirrus
      masdf1 = masir11 - masir12
c ... calculate secant of viewing zenith angle.
      cosvza = cos(vza*dtr)
      if (cosvza.ne.0.0) then
        schi = 1.0/cosvza
      else
        schi = 99.0
      end if

c ... interpolate look-up table values of 11 - 12 micron bt
c ... difference thresholds (function of viewing zenith
c ... and 11 micron brightness temperature).
      call tview(1,schi,masir11,diftemp)

c ... if a threshold was determined, then use this
c ... as the thin cirrus test, otherwise use a standard threshold
      if (diftemp.lt.0.1 .or. schi.eq.99.0) then
        dfthrsh = ds11_12hi(2)
```

```
      else
        dfthrsh = diftemp
      end if

c...  Set flag if test passed
      if (masdf1.le.dfthrsh) then
        nptests = nptests + 1
      end if
      loconf = dfthrsh + 0.5
      hiconf = dfthrsh - 0.5
      call conf_test(masdf1,loconf,hiconf,1.0,dfthrsh,1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      if (visusd) then
        if(masir11 .le. 320.0) then
          mas11_4 = masir11 - masir4
          if (mas11_4.ge.ds11_4lo(2) .or. mas11_4.le. ds11_4hi(2)) then
            nptests = nptests + 1
          end if
          locut(1) = ds11_4lo(1)
          locut(2) = ds11_4hi(1)
          hicut(1) = ds11_4lo(3)
          hicut(2) = ds11_4hi(3)
          midpt(1) = ds11_4lo(2)
          midpt(2) = ds11_4hi(2)

          call conf_test(mas11_4,locut,hicut,1.0,midpt,2,c3)
          cmin2 = min(cmin2,c3)
          ngtests(2) = ngtests(2) + 1
        endif
      end if

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******     END OF GROUP 2 TESTS ****************************
c
c
c
c ********  START OF GROUP 3 TESTS ****************************

      nptests = 0

c ... visible (.88 micron) reflectance threshold test.
      if (visusd) then
        if (masv88.le.dsref2(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv88,dsref2(1),dsref2(3),dsref2(4),
     +                 dsref2(2),1,c4)
        cmin3 = min(cmin3,c4)
        ngtests(3) = ngtests(3) + 1
      end if

      if(nptests .eq. ngtests(3) .and. ngtests(3) .ne. 0) then
        call set_bit(testbits,12)
      end if

c ******       END OF GROUP 3 TESTS    ****************************
c
c
c
c
c ***********    START OF GROUP 4 TESTS   ************************

      nptests = 0

c ... near infrared high cloud test
      if (visusd) then
        if (masv188.le.dsref3(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv188,dsref3(1),dsref3(3),dsref3(4),
     +                 dsref3(2),1,c6)
        cmin4 = min(cmin4,c6)
        ngtests(4) = ngtests(4) + 1
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ************    END OF GROUP 4 TESTS    ****************************

c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin3 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv66 .lt. dlref1(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```fortran
      subroutine LandDay_c(nbands,pxldat,avgtherm,vza,visusd,
     +                     vrused,eco_type,testbits,confdnc)

c F77 ***********************************************************
c
c      Routine for performing clear sky tests over coastal regions
c      during daylight hours.
c
c Input parameters:
c nbands          Number of MAS channels.
c pxldat          Array containing reflectance or brightness temperatures
c avgtherm        Average brightness temperature for given box
c                 for all bands for a single pixel
c vza             Current pixel viewing angle
c visusd          Logical variable indicating whether vis data used or not
c eco_type        Holder of ecosystem type (1-17)
c Output Parameters:
c testbits        two word 1-byte array containing bit results
c confdnc         product of all applied individual confidences
c
c END*************************************************************
c
      include 'thresholds.inc'

c ...
c ... scalar arguments ..
      real confdnc,vza
      integer nbands
      logical visusd,vrused
      byte eco_type
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,c5,cosvza,dfthrsh,diftemp,dtr,mas11_4,masdf1,
     +     masir11,masir12,masir13,masir4,masir8,masv162,masv188,
     +     masir3,masv95,masv66,masv88,pi,schi,vrat,c6,pre_confdnc,
     +     groups,fac
      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2),midpt(2)
      integer ngtests(4)
c ...
c ... external subroutines ..
      external conf_test,tview,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied in each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... place band values into individual variables for easy
c ... identification
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv95 = pxldat(4)
      masv162 = pxldat(5)
      masv188 = pxldat(6)
      masir3 = pxldat(8)
      masir4 = pxldat(9)
      masir8 = pxldat(10)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... the "cmin" variables represent test group confidences
      cmin1 = 1.0
      cmin2 = 1.0
      cmin3 = 1.0
      cmin4 = 1.0

c ... perform tests.
c
c      **** GROUP 1 TESTS ************************************

      nptests = 0

c ... co2 high cloud test
      if (masir13.gt.dlco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,dlco2(1),dlco2(3),dlco2(4),
     *               dlco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c      *****  END OF GROUP 1 TESTS  **************************
c
c
c
c      ****  GROUP 2 TESTS  **********************************

      nptests = 0

c ... 11-12um brightness temperature difference test (APOLLO test)
c ... for thin cirrus).
      masdf1 = masir11 - masir12
c ... calculate secant of viewing zenith angle.
      cosvza = cos(vza*dtr)
      if (cosvza.ne.0.0) then
        schi = 1.0/cosvza
      else
        schi = 99.0
      end if

c ... interpolate look-up table values of 11 - 12 micron bt
c ... difference thresholds (function of viewing zenith
c ... and 11 micron brightness temperature).
      call tview(1,schi,masir11,diftemp)

c ... if a valid threshold was determined, then use this
c ... value, otherwise use the standard threshold
      if (diftemp.lt.0.1 .or. schi.eq.99.0) then
```

```
        dfthrsh = dl11_12hi(1)
      else
        dfthrsh = diftemp
      end if

      if (masdf1.le.dfthrsh) then
        nptests = nptests + 1
      end if
      locut(1) = dfthrsh + 0.5
      hicut(1) = dfthrsh - 1.25
      call conf_test(masdf1,locut,hicut,1.0,dfthrsh,1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      if (visusd) then
        if(masir4.ne.32767.0.and.masir11.ne.32767.0) then
          mas11_4 = masir11 - masir4
          if (mas11_4.ge.dl11_4lo(2)) then
            nptests = nptests + 1
          end if
          call conf_test(mas11_4,dl11_4lo(1),dl11_4lo(3),dl11_4lo(4),
     +                   dl11_4lo(2),1,c3)
          cmin2 = min(cmin2,c3)
          ngtests(2) = ngtests(2) + 1
        end if
      end if

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c ******      END OF GROUP 2 TESTS ***************************
c
c
c
c *******  START OF GROUP 3 TESTS ***************************

      nptests = 0

c ... visible reflectance threshold test.
      if (visusd) then
        if (masv66.le.dlref1(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv66,dlref1(1),dlref1(3),dlref1(4),
     +                 dlref1(2),1,c4)
        cmin3 = min(cmin3,c4)
        ngtests(3) = ngtests(3) + 1
      end if

      if(nptests .eq. ngtests(3) .and. ngtests(3) .ne. 0) then
        call set_bit(testbits,12)
      end if

c ******       END OF GROUP 3 TESTS    ***************************
c
c
c
c
c ***********   START OF GROUP 4 TESTS   ************************

      nptests = 0

c ... near infrared high cloud test
      if(visusd) then
        if (masv188.le.dlref3(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv188,dlref3(1),dlref3(3),dlref3(4),
     +                 dlref3(2),1,c6)
        cmin4 = min(cmin4,c6)
        ngtests(4) = ngtests(4) + 1
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ***********    END OF GROUP 4 TESTS    ***************************
c
c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin3 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. dlref3(2) .and. masv188 .ge. dlref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv66 .lt. dlref1(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```
      subroutine LandDay_c(nbands,pxldat,avgtherm,vza,visusd,
     +                     vrused,eco_type,testbits,confdnc)

c F77 ************************************************************
c
c      Routine for performing clear sky tests over coastal regions
c      during daylight hours.
c
c Input parameters:
c nbands          Number of MAS channels.
c pxldat          Array containing reflectance or brightness temperatures
c avgtherm        Average brightness temperature for given box
c                 for all bands for a single pixel
c vza             Current pixel viewing angle
c visusd          Logical variable indicating whether vis data used or not
c eco type        Holder of ecosystem type (1-17)
c output Parameters:
c testbits        two word 1 byte array containing bit results
c confdnc         product of all applied individual confidences
c
c END*************************************************************
c
      include 'thresholds.inc'

c ...
c ... scalar arguments ..
      real confdnc,vza
      integer nbands
      logical visusd,vrused
      byte eco_type
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,c5,cosvza,dfthrsh,diftemp,dtr,masl1_4,masdf1,
     +     masir11,masir12,masir13,masir4,masir8,masv162,masv188,
     +     masir3,masv95,masv66,masv88,pi,schi,vrat,c6,pre_confdnc,
     +     groups,fac
      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2),midpt(2)
      integer ngtests(4)
c ...
c ... external subroutines ..
      external conf_test,tview,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied in each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... place band values into individual variables for easy
c ... identification
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv95 = pxldat(4)
      masv162 = pxldat(5)
      masv188 = pxldat(6)
      masir3 = pxldat(8)
      masir4 = pxldat(9)
      masir8 = pxldat(10)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... the "cmin" variables represent test group confidences
      cmin1 = 1.0
      cmin2 = 1.0
      cmin3 = 1.0
      cmin4 = 1.0

c ... perform tests.
c
c      **** GROUP 1 TESTS ************************************

      nptests = 0

c ... co2 high cloud test
      if (masir13.gt.dlco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,dlco2(1),dlco2(3),dlco2(4),
     *               dlco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c      *****  END OF GROUP 1 TESTS  *************************
c
c
c
c      ****  GROUP 2 TESTS  **********************************

      nptests = 0

c ... 11-12um brightness temperature difference test (APOLLO test)
c ... for thin cirrus).
      masdf1 = masir11 - masir12
c ... calculate secant of viewing zenith angle.
      cosvza = cos(vza*dtr)
      if (cosvza.ne.0.0) then
        schi = 1.0/cosvza
      else
        schi = 99.0
      end if

c ... interpolate look-up table values of 11 - 12 micron bt
c ... difference thresholds (function of viewing zenith
c ... and 11 micron brightness temperature).
      call tview(1,schi,masir11,diftemp)

c ... if a valid threshold was determined, then use this
c ... value, otherwise use the standard threshold
      if (diftemp.lt.0.1 .or. schi.eq.99.0) then
```

```
          dfthrsh = dl11_12hi(1)
        else
          dfthrsh = difftemp
        end if

        if (masdf1.le.dfthrsh) then
          nptests = nptests + 1
        end if
        locut(1) = dfthrsh + 0.5
        hicut(1) = dfthrsh - 1.25
        call conf_test(masdf1,locut,hicut,1.0,dfthrsh,1,c2)
        cmin2 = min(cmin2,c2)
        ngtests(2) = ngtests(2) + 1

c ... 11 minus 4 micron BTDIF fog and low cloud test.
        if (visusd) then
          if(masir4.ne.32767.0.and.masir11.ne.32767.0) then
            mas11_4 = masir11 - masir4
            if (mas11_4.ge.dl11_4lo(2)) then
              nptests = nptests + 1
            end if
            call conf_test(mas11_4,dl11_4lo(1),dl11_4lo(3),dl11_4lo(4),
     +                     dl11_4lo(2),1,c3)
            cmin2 = min(cmin2,c3)
            ngtests(2) = ngtests(2) + 1
          end if
        end if

        if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
          call set_bit(testbits,11)
        end if

c *******     END OF GROUP 2 TESTS ***************************
c
c
c
c ********  START OF GROUP 3 TESTS ***************************

        nptests = 0

c ... visible reflectance threshold test.
        if (visusd) then
          if (masv66.le.dlref1(2)) then
            nptests = nptests + 1
          end if
          call conf_test(masv66,dlref1(1),dlref1(3),dlref1(4),
     +                   dlref1(2),1,c4)
          cmin3 = min(cmin3,c4)
          ngtests(3) = ngtests(3) + 1
        end if

        if(nptests .eq. ngtests(3) .and. ngtests(3) .ne. 0) then
          call set_bit(testbits,12)
        end if

c ******      END OF GROUP 3 TESTS    ***************************
c
c
c
c
c ***********   START OF GROUP 4 TESTS   *************************

        nptests = 0

c ... near infrared high cloud test
        if(visusd) then
          if (masv188.le.dlref3(2)) then
            nptests = nptests + 1
          end if
          call conf_test(masv188,dlref3(1),dlref3(3),dlref3(4),
     +                   dlref3(2),1,c6)
          cmin4 = min(cmin4,c6)
          ngtests(4) = ngtests(4) + 1
        end if

        if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
          call set_bit(testbits,13)
        end if

c ***********    END OF GROUP 4 TESTS    ***************************
c
c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin3 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv66 .lt. dlref1(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```
      subroutine ocean_day(nbands,pxldat,avgtherm,vza,pxrfa,
     +                     snglnt,visusd,eco_type,pw,
     +                     testbits,confdnc)
c F77 ***********************************************************
c
c      Routine for performing clear sky tests over water
c      surfaces during daylight hours.
c
c Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c vza           Current pixel viewing angle
c snglnt        Logical variable flagging sunglint pixels
c visusd        Logical variable indicating whether vis data used or not
c eco_type      Holder of ecosystem type (1-17)
c pw            Amount of precipitable water at pixel site
c avgtherm      Average BT over box of pixels
c pxrfa         reflectance angle used for determining sunglint
c Output Parameters:
c testbits      two word 1-byte array containing bit results
c confdnc       product of all applied individual confidences
c
c END*************************************************************
c
      INCLUDE 'thresholds.inc'
c ...
c ... scalar arguments ..
      real confdnc,pw,vza
      integer nbands
      logical snglnt,visusd
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,c5,c6,c7,cosvza,ct1,ct2,dfthrsh,diftemp,diftsp1,
     +     diftsp2,dtr,mas11_4,masdf1,masdf2,masir11,masir12,masir13,
     +     masir4,masir8,masir3,masv188,masv95,masv66,masv88,
     +     pi,schi,vrat,masir65,c8,c9,c10,
     +     c11,mas_swir,groups,fac,pre_confdnc
      integer nptests,ngtests(4)
      byte eco_type
c ...
c ... local arrays ..
      real hicut(2),locut(2),midpt(2)
c ...
c ... external functions ..
      real rega,regb
      external rega,regb
c ...
c ... external subroutines ..
      external conf_test,tview,clear_bit,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos,cos
c ...
c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... set confidence to 1.0 to begin with
      confdnc = 1.0

c ... Place band values into individual variables for easy
c ... identification.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv95 = pxldat(4)
      masv188 = pxldat(6)
      masir3 = pxldat(8)
      masir4 = pxldat(9)
      masir8 = pxldat(10)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... The "cmin" variables represent test group confidences.
      cmin1 = 1.0
      cmin2 = 1.0
      cmin3 = 1.0
      cmin4 = 1.0

C     **** GROUP 1 TESTS *************************************

      nptests = 0

c     11 micron brigthness temperature threshold test
c ... compare to daytime ocean threshold, set bit if passed
      if (masir11 .ge. dobt11(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir11,dobt11(1),dobt11(3),dobt11(4),
     *               dobt11(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

c ... co2 high cloud test
      if (masir13.gt.doco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,doco2(1),doco2(3),doco2(4),
     *              doco2(2),1,c2)
      cmin1 = min(cmin1,c2)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  **************************
c
c
c     ****. GROUP 2 TESTS  **********************************

      nptests = 0

c ... Tri-spectral tests - apply only if a pw value exists for this
c ... pixel.
      if (pw .gt. 1.0) then
```

```
c ...   get dynamic clear sky thresholds based on pw
        diftsp2 = rega(pw)
        diftsp1 = regb(pw)
c ...   calculate 8 minus 11 and 11 minus 12 micron BTDIFs
        masdf2 = masir8 - masir11
        masdf1 = masir11 - masir12
c ...   if both BTDIF's pass the tests, then set bit
        if ((masdf2.lt.diftsp2) .and. (masdf1.gt.diftsp1)) then
          nptests = nptests + 1
        end if
        locut(1) = diftsp2 + .5
        hicut(1) = diftsp2 - .5
        call conf_test(masdf2,locut,hicut,1.0,diftsp2,1,ct1)
        locut(1) = diftsp1 - .5
        hicut(1) = diftsp1 + .5
        call conf_test(masdf1,locut,hicut,1.0,diftsp1,1,ct2)
        c3 = ct1*ct2
        cmin2 = min(cmin2,c3)
        ngtests(2) = ngtests(2) + 1
        c4 = 0.0

      else

c ...   If no pw value exists for this pixel, use the low cloud
c ...   11-12 micron BTDIF test.

        masdf1 = masir11 - masir12
        if (masdf1.gt.do11_12lo(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masdf1,do11_12lo(1),do11_12lo(3),do11_12lo(4),
     +                 do11_12lo(2),1,c4)
        cmin2 = min(cmin2,c4)
        ngtests(2) = ngtests(2) + 1
        c3 = 0.0

      end if

c ... 11-12um brightness temperature difference test
c ... for thin cirrus.
c ... calculate secant of viewing zenith angle.
      cosvza = cos(vza*dtr)
      if (cosvza.ne.0.0) then
        schi = 1.0/cosvza
      else
        schi = 99.0
      end if

c ... Interpolate look-up table values of 11 - 12 micron bt
c ... difference thresholds (function of viewing zenith
c ... and 11 micron brightness temperature).
      call tview(1,schi,masir11,diftemp)

c ... If a threshold was determined, then use this value
c ... as the test threshold, otherwise use the standard threshold.
      if (diftemp.lt.0.1 .or. schi.eq.99.0) then
        dfthrsh = do11_12hi(1)
      else
        dfthrsh = diftemp
      end if

      if (masdf1.le.dfthrsh) then
        nptests = nptests + 1
      end if
      locut(1) = dfthrsh + 0.5
      hicut(1) = dfthrsh - 1.25
      call conf_test(masdf1,locut,hicut,1.0,dfthrsh,1,c5)
      cmin2 = min(cmin2,c5)
      ngtests(2) = ngtests(2) + 1

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      if (visusd) then
        mas11_4 = masir11 - masir4
        if (mas11_4.ge.do11_4lo(2)) then
          nptests = nptests + 1
        end if
        call conf_test(mas11_4,do11_4lo(1),do11_4lo(3),do11_4lo(4),
     +                 do11_4lo(2),1,c6)
        cmin2 = min(cmin2,c6)
        ngtests(2) = ngtests(2) + 1
      end if

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******     END OF GROUP 2 TESTS ****************************
c
c
c ********  START OF GROUP 3 TESTS ****************************

      nptests = 0

c ... visible reflectance threshold test.

      if (visusd) then

        if(snglnt) then
          ibin = int(pxrfa) + 1
          if(ibin .gt. 36) ibin = 36
          locut(1) = snglnt2cl(ibin,1)
          hicut(1) = snglnt2ch(ibin,1)
          midpt(1) = snglnt2(ibin,1)
        else
          locut(1) = doref2(1)
          hicut(1) = doref2(3)
          midpt(1) = doref2(2)
        end if

        if (masv88.le.midpt(1)) then
          nptests = nptests + 1
        end if
        call conf_test(masv88,locut(1),hicut(1),doref2(4),
     +                 midpt(1),1,c7)
        cmin3 = min(cmin3,c7)
        ngtests(3) = ngtests(3) + 1

      end if

c ... visible channel ratio test (channel 2 / channel 1)

      if (visusd) then

        if(snglnt) then
          ibin = int(pxrfa) + 1
          if(ibin .gt. 36) ibin = 36
          locut(1) = snglntvcl(ibin,1)
          locut(2) = snglntvcl(ibin,2)
          hicut(1) = snglntvch(ibin,1)
```

```
          hicut(2) = snglntvch(ibin,2)
          midpt(1) = snglntv(ibin,1)
          midpt(2) = snglntv(ibin,2)
        else
          locut(1) = dovratlo(1)
          locut(2) = dovrathi(1)
          hicut(1) = dovratlo(3)
          hicut(2) = dovrathi(3)
          midpt(1) = dovratlo(2)
          midpt(2) = dovrathi(2)
        end if

        vrat = masv88/masv66
        if (vrat.lt.midpt(1) .or. vrat.gt.midpt(2)) then
          nptests = nptests + 1
        end if
        call conf_test(vrat,locut,hicut,1.0,midpt,2,c8)
        cmin3 = min(cmin3,c8)
        ngtests(3) = ngtests(3) + 1

      end if

      if(nptests .eq. ngtests(3) .and. ngtests(3) .ne. 0) then
        call set_bit(testbits,12)
      end if

c ******        END OF GROUP 3 TESTS    ****************************
c
c
c ***********    START OF GROUP 4 TESTS   *************************

      nptests = 0

c ... Near-infrared high cloud test.
      if (visusd .and. .not. snglnt) then
        if (masv188.le.doref3(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv188,doref3(1),doref3(3),doref3(4),
     +                 doref3(2),1,c11)
        cmin4 = min(cmin4,c11)
        ngtests(4) = ngtests(4) + 1
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ************    END OF GROUP 4 TESTS    ****************************
c
c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin3 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv88 .lt. doref2(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```fortran
      subroutine DesertDay(nbands,pxldat,vza,visusd,avgtherm,
     +                     testbits,confdnc)

c F77 ************************************************************
c
c Description:
c      Routine for performing clear sky threshold tests over desert
c      surfaces during daylight hours.
c
c Input parameters:
c nbands         Number of MAS channels
c pxldat         Array containing reflectance or brightness temperatures
c                for all bands for a single pixel
c avgtherm       Average BT over region
c vza            Current pixel viewing angle
c visusd         Logical variable indicating whether vis data used or not
c
c Output Parameters:
c testbits       two word 1-byte array containing bit results
c confdnc        product of all applied individual confidences
c
c End-----------------------------------------------------------------

      include 'thresholds.inc'

c ... scalar arguments ..
      integer nbands
      real confdnc,vza
      logical visusd
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,c5,cosvza,dfthrsh,diftemp,dtr,mas11_4,masdf1,
     +     masir11,masir12,masir13,masir4,masv188,masv66,masv88,
     +     pi,schi,vrat,c6,cmin1,cmin2,cmin3,cmin4,hiconf,loconf,
     +     fac,pre_confdnc,groups

      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2),midpt(2)
      integer ngtests(4)
c ...
c ... external subroutines ..
      external conf_test,tview,set_bit,clear_bit
c ...
c ... intrinsic functions ..
      intrinsic cos,acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... set confidence to 1.0 to begin with
      confdnc = 1.0

c ... Place band values into individual variables for easy
c ... identification.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv188 = pxldat(6)
      masir4 = pxldat(9)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... The "cmin" variables represent group test confidences.
      cmin1 = 1.0
      cmin2 = 1.0
      cmin3 = 1.0
      cmin4 = 1.0

c ... Perform tests.
c
C     **** GROUP 1 TESTS ************************************

      nptests = 0

c ... Co2 high cloud test.
      if (masir13.gt.dsco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,dsco2(1),dsco2(3),dsco2(4),
     *               dsco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  **************************
c
c
c
c     ****  GROUP 2 TESTS  **********************************

      nptests = 0

c ... 11-12um brightness temperature difference test
c ... for thin cirrus.
      masdf1 = masir11 - masir12
c ... calculate secant of viewing zenith angle.
      cosvza = cos(vza*dtr)
      if (cosvza.ne.0.0) then
        schi = 1.0/cosvza
      else
        schi = 99.0
      end if

c ... Interpolate look-up table values of 11 - 12 micron bt
c ... difference thresholds (function of viewing zenith
c ... and 11 micron brightness temperature).
      call tview(1,schi,masir11,diftemp)

c ... If a threshold was determined, then use this
c ... as the thin cirrus test, otherwise use a standard threshold.
      if (diftemp.lt.0.1 .or. schi.eq.99.0) then
        dfthrsh = ds11_12hi(2)
```

```
      else
        dfthrsh = diftemp
      end if

      if (masdf1.le.dfthrsh) then
        nptests = nptests + 1
      end if
      loconf = dfthrsh + 0.5
      hiconf = dfthrsh - 1.25
      call conf_test(masdf1,loconf,hiconf,1.0,dfthrsh,1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      if (visusd) then
        if(masir11 .le. 320.0) then
          mas11_4 = masir11 - masir4
          if (mas11_4.ge.ds11_4lo(2) .or. mas11_4.le. ds11_4hi(2)) then
            nptests = nptests + 1
          end if
          locut(1) = ds11_4lo(1)
          locut(2) = ds11_4hi(1)
          hicut(1) = ds11_4lo(3)
          hicut(2) = ds11_4hi(3)
          midpt(1) = ds11_4lo(2)
          midpt(2) = ds11_4hi(2)

          call conf_test(mas11_4,locut,hicut,1.0,midpt,2,c3)
          cmin2 = min(cmin2,c3)
          ngtests(2) = ngtests(2) + 1
        endif
      end if

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******     END OF GROUP 2 TESTS ***************************
c
c
c
c ********  START OF GROUP 3 TESTS ***************************

      nptests = 0

c ... visible (.88 micron) reflectance threshold test.
      if (visusd) then
        if (masv88.le.dsref2(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv88,dsref2(1),dsref2(3),dsref2(4),
     +                 dsref2(2),1,c4)
        cmin3 = min(cmin3,c4)
        ngtests(3) = ngtests(3) + 1
      end if

      if(nptests .eq. ngtests(3) .and. ngtests(3) .ne. 0) then
        call set_bit(testbits,12)
      end if

c ******       END OF GROUP 3 TESTS    ***************************
c
c
c
c

c ***********    START OF GROUP 4 TESTS   ************************

      nptests = 0

c ... near-infrared high cloud test
      if (visusd) then
        if (masv188.le.dsref3(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv188,dsref3(1),dsref3(3),dsref3(4),
     +                 dsref3(2),1,c6)
        cmin4 = min(cmin4,c6)
        ngtests(4) = ngtests(4) + 1
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ************    END OF GROUP 4 TESTS    ***************************
c
c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin3 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv66 .lt. dlref1(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```
      subroutine DesertDay(nbands,pxldat,vza,visusd,avgtherm,
     +                     testbits,confdnc)

c F77 *****************************************************************
c
c Description:
c     Routine for performing clear sky threshold tests over desert
c     surfaces during daylight hours.
c
c Input parameters:
c nbands        Number of MAS channels
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c avgtherm      Average BT over region
c vza           Current pixel viewing angle
c visusd        Logical variable indicating whether vis data used or not
c
c Output Parameters:
c testbits      two word 1-byte array containing bit results
c confdnc       product of all applied individual confidences
c
c End-----------------------------------------------------------------

      include 'thresholds.inc'

c ... scalar arguments ..
      integer nbands
      real confdnc,vza
      logical visusd
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,c5,cosvza,dfthrsh,diftemp,dtr,mas11_4,masdf1,
     +     masir11,masir12,masir13,masir4,masv188,masv66,masv88,
     +     pi,schi,vrat,c6,cmin1,cmin2,cmin3,cmin4,hiconf,loconf,
     +     fac,pre_confdnc,groups

      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2),midpt(2)
      integer ngtests(4)
c ...
c ... external subroutines ..
      external conf_test,tview,set_bit,clear_bit
c ...
c ... intrinsic functions ..
      intrinsic cos,acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... set confidence to 1.0 to begin with
      confdnc = 1.0

c ... Place band values into individual variables for easy
c ... identification.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv188 = pxldat(6)
      masir4 = pxldat(9)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... The "cmin" variables represent group test confidences.
      cmin1 = 1.0
      cmin2 = 1.0
      cmin3 = 1.0
      cmin4 = 1.0

c ... Perform tests.
c
C     **** GROUP 1 TESTS ***********************************

      nptests = 0

c ... Co2 high cloud test.
      if (masir13.gt.dsco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,dsco2(1),dsco2(3),dsco2(4),
     *               dsco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  **************************
c
c
c
c     ****  GROUP 2 TESTS  **********************************

      nptests = 0

c ... 11-12um brightness temperature difference test
c ... for thin cirrus.
      masdf1 = masir11 - masir12
c ... calculate secant of viewing zenith angle.
      cosvza = cos(vza*dtr)
      if (cosvza.ne.0.0) then
        schi = 1.0/cosvza
      else
        schi = 99.0
      end if

c ... Interpolate look-up table values of 11 - 12 micron bt
c ... difference thresholds (function of viewing zenith
c ... and 11 micron brightness temperature).
      call tview(1,schi,masir11,diftemp)

c ... If a threshold was determined, then use this
c ... as the thin cirrus test, otherwise use a standard threshold.
      if (diftemp.lt.0.1 .or. schi.eq.99.0) then
        dfthrsh = ds11_12hi(2)
```

```fortran
      else
        dfthrsh = diftemp
      end if

      if (masdf1.le.dfthrsh) then
        nptests = nptests + 1
      end if
      loconf = dfthrsh + 0.5
      hiconf = dfthrsh - 1.25
      call conf_test(masdf1,loconf,hiconf,1.0,dfthrsh,1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      if (visusd) then
        if(masir11 .le. 320.0) then
          mas11_4 = masir11 - masir4
          if (mas11_4.ge.ds11_4lo(2) .or. mas11_4.le. ds11_4hi(2)) then
            nptests = nptests + 1
          end if
          locut(1) = ds11_4lo(1)
          locut(2) = ds11_4hi(1)
          hicut(1) = ds11_4lo(3)
          hicut(2) = ds11_4hi(3)
          midpt(1) = ds11_4lo(2)
          midpt(2) = ds11_4hi(2)

          call conf_test(mas11_4,locut,hicut,1.0,midpt,2,c3)
          cmin2 = min(cmin2,c3)
          ngtests(2) = ngtests(2) + 1
        endif
      end if

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******      END OF GROUP 2 TESTS ***************************
c
c
c
c ********  START OF GROUP 3 TESTS ***************************

      nptests = 0

c ... visible (.88 micron) reflectance threshold test.
      if (visusd) then
        if (masv88.le.dsref2(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv88,dsref2(1),dsref2(3),dsref2(4),
     +                 dsref2(2),1,c4)
        cmin3 = min(cmin3,c4)
        ngtests(3) = ngtests(3) + 1
      end if

      if(nptests .eq. ngtests(3) .and. ngtests(3) .ne. 0) then
        call set_bit(testbits,12)
      end if

c ******       END OF GROUP 3 TESTS    ****************************
c
c
c
c
c ***********    START OF GROUP 4 TESTS    ************************

      nptests = 0

c ... near-infrared high cloud test
      if (visusd) then
        if (masv188.le.dsref3(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv188,dsref3(1),dsref3(3),dsref3(4),
     +                 dsref3(2),1,c6)
        cmin4 = min(cmin4,c6)
        ngtests(4) = ngtests(4) + 1
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ************    END OF GROUP 4 TESTS    ***************************
c
c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin3 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv66 .lt. dlref1(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```
      subroutine clear_bit(testbits,bit_num)

c     Routine for clearing a single bit within the 48-bit
c     cloud mask output array.

      integer*4 bit_num
      byte testbits(2)

c     Determine which word (1-2) of the 1-byte array contains the
c     bit of interest.

      iword = (bit_num / 8) + 1

c     Determine the position of the bit within the current
c     8-bit segment (1-byte word).

      ipos = bit_num - ((iword-1) * 8)

      itest = testbits(iword)
      itest = ibclr(itest,ipos)

      testbits(iword) = itest

c     write(*,'(1x,''clear '',4i10)') iword,ipos,itest,testbits(iword)

      return
      end
```

```
      subroutine set_bit(testbits,bit_num)

c     Routine for setting a single bit within the 48-bit
c     cloud mask output array.

      integer*4 bit_num
      byte testbits(2)

c     Determine which word (1-2) of the 1-byte array contains the
c     bit of interest.

      iword = (bit_num / 8) + 1

c     Determine the position of the bit within the current
c     8-bit segment (1-byte word).

      ipos = bit_num - ((iword-1) * 8)

      itest = testbits(iword)
      itest = ibset(itest,ipos)

      testbits(iword) = itest

c     write(*,'(1x,''set '',4i10)') iword,ipos,itest,testbits(iword)

      return
      end
```

```
      subroutine get_refl3(tbb3,tbb4,jday,sza,ref3)

c     Routine for calculating the reflectance of the
c     solar component of the MAS channel 3.75 um radiance.

      parameter(pi=3.14159)
      parameter(dtr=pi/180.0)
      parameter(c1=1.191e-16)
      parameter(c2=1.4388e-02)
      parameter(wvnm3=2671.65)
      parameter(wvnm4=907.28)
      parameter(wvln3p=10000.0/(wvnm3*1.0e+06))
      parameter(wvln3=10000.0/wvnm3)

      real*4 tbb3,tbb4,solr_irrad

c     Calculate thermal component of channel 3 radiance
c     based on channel 4 brightness temperature.

c     thr3 = (c1*(wvnm3**3)) / (exp(c2*wvnm3/tbb4)-1.0)
      thr3 = c1 / ( (wvln3p**5.0) * (exp(c2/(wvln3p*tbb4)) - 1.0) )
      thr3 = thr3 * 1.0e-06

c     Calculate total radiance of channel 3.

c     tot3 = (c1*(wvnm3**3)) / (exp(c2*wvnm3/tbb3)-1.0)
      tot3 = c1 / ( (wvln3p**5.0) * (exp(c2/(wvln3p*tbb3)) - 1.0) )
      tot3 = tot3 * 1.0e-06

c     Calculate solar irradiance for channel 3.

      solr = solr_irrad(wvln3,jday)
      solr = solr * cos(sza*dtr)

c     Get reflectance.

      sol3 = tot3 - thr3
      ref3 = ((sol3 * pi) / solr) * 100.0


      return
      end
```

```fortran
      subroutine conf_test(val,locut,hicut,power,midpt,nmval,
     *                     conflev)

c     Routine for determining the level of confidence of a
c     particular clear sky test.  Input single threshold value
c     ('midpt') with associated confidence limits ('locut', 'hicut')
c     or two threshold values and associated limits which define
c     a range of values for a test.  "Low" and "high" cutoffs
c     refer to low or high confidence ends of an interval and
c     not necessarily to absolute value.  Routine calculates the
c     confidence based on an "S" function.  One may change the shape
c     of the function by changing 'power' and/or 'midpt'.

      integer*4 nmval
      real*4 alpha,gamma,power,val,locut(2),hicut(2),range,
     *       coeff,s1,midpt(2),c
      logical*4 flipped

      coeff = 2.0 ** (power - 1.0)

c     Check if testing a single threshold or a range of values.

      if(nmval .eq. 1) then

c        Single threshold.

         if(hicut(1) .gt. locut(1)) then
            gamma = hicut(1)
            alpha = locut(1)
            flipped = .false.
         else
            gamma = locut(1)
            alpha = hicut(1)
            flipped = .true.
         end if
         beta = midpt(1)

c        Check for value beyond function range.

         if(.not. flipped .and. val .gt. gamma) then
            c = 1.0

         else if(.not. flipped .and. val .lt. alpha) then
            c = 0.0

         else if(flipped .and. val .gt. gamma) then
            c = 0.0

         else if(flipped .and. val .lt. alpha) then
            c = 1.0

         else

c           Value is within the range of the function.

            if(val .le. beta) then

               range = 2.0 * (beta - alpha)
               s1 = (val - alpha) / range
               if(.not.flipped) c = coeff * s1**power
               if(flipped) c = 1.0 - (coeff * s1**power)

            else

               range = 2.0 * (beta - gamma)
               s1 = (val - gamma) / range
               if(.not.flipped) c = 1.0 - (coeff * s1**power)
               if(flipped) c = coeff * s1**power

            end if

         end if

      else if(nmval .eq. 2) then

c        Range.

         gamma1 = hicut(1)
         gamma2 = hicut(2)
         alpha1 = locut(1)
         alpha2 = locut(2)
         beta1 = midpt(1)
         beta2 = midpt(2)

c        Find if interval between inner cutoffs passes test
c        or fails.

         if( (alpha1-gamma1) .gt. 0.0) then

c           Inner region fails test.
c           Check for value beyond function range.

            if(val .gt. alpha1 .and. val .lt. alpha2) then

               c = 0.0

            else if(val .lt. gamma1 .or. val .gt. gamma2) then

               c = 1.0

            else if(val .le. alpha1) then

c              Value is within range of lower set of limits.

               if(val .ge. beta1) then
                  range = 2.0 * (beta1 - alpha1)
                  s1 = (val - alpha1) / range
                  c = coeff * s1**power
               else
                  range = 2.0 * (beta1 - gamma1)
                  s1 = abs(val - gamma1) / range
                  c = 1.0 - (coeff * s1**power)
               end if

            else

c              Value is within range of upper set of limits.

               if(val .le. beta2) then
                  range = 2.0 * (beta2 - alpha2)
                  s1 = (val - alpha2) / range
                  c = coeff * s1**power
               else
                  range = 2.0 * (beta2 - gamma2)
                  s1 = (val - gamma2) / range
                  c = 1.0 - (coeff * s1**power)
               end if

            end if
```

```
         else

c           Inner region passes test.
c           Check for value beyond function range.

            if(val .gt. gamma1 .and. val .lt. gamma2) then

               c = 1.0

            else if(val .lt. alpha1 .or. val .gt. alpha2) then

               c = 0.0

            else if(val .le. gamma1) then

c              Value is within range of lower set of limits.

               if(val .le. beta1) then
                  range = 2.0 * (beta1 - alpha1)
                  s1 = (val - alpha1) / range
                  c = coeff * s1**power
               else
                  range = abs(2.0 * (beta1 - gamma1))
                  s1 = abs((val - gamma1) / range)
                  c = 1.0 - (coeff * s1**power)
               end if

            else

c              Value is within range of upper set of limits.

               if(val .ge. beta2) then
                  range = 2.0 * (beta2 - alpha2)
                  s1 = (val - alpha2) / range
                  c = coeff * s1**power
               else
                  range = 2.0 * (beta2 - gamma2)
                  s1 = (val - gamma2) / range
                  c = 1.0 - (coeff * s1**power)
               end if

            end if

         end if

      else

         write(*,'(1x,''Invalid number of thresholds'')')
         return

      end if

c     Force confidence values to be between 0 and 1.

      if(c .gt. 1.0) c = 1.0
      if(c .lt. 0.0) c = 0.0

      conflev = c

      return
      end
```

```
      real function rega(pw)

c F77 ************************************************************
c ...
c ... regression function relating the pw values to the 8-11um btdif.  the
c ... difference is related to the amount of pw in the atmosphere
c ... due to the weak water vapor lines present in this spectral
c ... region.  the regressions were determined from actual hirs
c ... data, and tweaked for use with the mas bandwidths.  this
c ... information provides a threshold value for 8-11um clear sky
c ... determination.
c ...
c Input parameters:
c pw - total column precipitable water value (g/cm-2)
c Output Parameters:
c rega - 8-11um clear sky threshold
c
c END**************************************************************
c
c ...
c ... scalar arguments ..
      real pw
c ...
c ... local scalars ..
c ... variables: a - slope of regression
c ... b - intercept of regression
      real a,b
c ...
c ... intrinsic functions ..
      intrinsic log
c ...
c ... data statements ..
c ... coefficients for mas data
      data a/-1.64805/,b/-3.19767/
c ...
c ...
      rega = (a*log(pw)) + b
c ...
      return
      end
```

```
      real function regb(pw)

c F77 ****************************************************************
c ...
c ... regression function relating the pw values to the 11-12um btdif.  the
c ... difference is related to the amount of pw in the atmosphere
c ... due to the weak water vapor lines present in this spectral
c ... region.  the regressions were determined from actual hirs
c ... data, and tweaked for use with the mas bandwidths.  this
c ... information provides a threshold value for 11-12um clear sky
c ... determination.
c ...
c Input parameters:
c pw - total column precipitable water value (g/cm-2)
c Output Parameters:
c regb - 8-11um clear sky threshold
c
c END*******************************************************************

c ...
c ... scalar arguments ..
      real pw
c ...
c ... local scalars ..
c ... variables: c - slope of regression
c ... d - intercept of regression
      real c,d
c ...
c ... data statements ..
c ... coefficients for mas data
      data c/0.488198/,d/-0.456924/
c ...
c ...
      regb = (c*pw) + d
c ...
      return
      end
```

```
      SUBROUTINE TVIEW(KEY,XMU,BT11,CORR)
C
C
C     BI-DIMENSIONAL LINEAR OR QUADRATIC INTERPOLATION SCHEME ( LAGRANGE FORM )
C
C     INPUT PARAMETERS
C
C     KEY = 1 FOR LINEAR INTERPOLATION
C         = 2 FOR QUADRATIC INTERPOLATION
C     XMU = SECANT OF VIEWING ANGLE
C     BT11 = CHANNEL 4 BT
C
C     OUTPUT PARAMETERS
C
C     CORR = CORRECTION FACTOR FROM APOLLO
C
      REAL LU0,LU1,LU2,LT0,LT1,LT2
      DIMENSION UTAB(5),TTAB(6),TAB(5,6)
      DATA UTAB/2.00,1.75,1.5,1.25,1.00/
      DATA TTAB/260.,270.,280.,290.,300.,310./

      DATA (TAB(I, 1),I=1,5)/
     * 1.10,0.90,0.65,0.60,0.55/
      DATA (TAB(I, 2),I=1,5)/
     * 1.13,1.03,0.81,0.63,0.58/
      DATA (TAB(I, 3),I=1,5)/
     * 2.30,2.14,1.88,1.61,1.30/
      DATA (TAB(I, 4),I=1,5)/
     * 4.73,4.27,3.95,3.72,3.06/
      DATA (TAB(I, 5),I=1,5)/
     * 8.43,7.42,7.00,6.92,5.77/
      DATA (TAB(I, 6),I=1,5)/
     * 13.39,11.60,11.03,10.74,9.41/


C
C     BOUNDS CHECK
C
      U=XMU
      T=BT11
      IF(U.GT.UTAB(1))U=UTAB(1)
      IF(U.LT.UTAB(5))U=UTAB(5)
      IF(T.LT.TTAB(1))T=TTAB(1)
      IF(T.GT.TTAB(6))T=TTAB(6)
C
C     SELECT THE XMU INDICIES
C
      DO 1 I=2,5
      II=I
      IF(U.GE.UTAB(I))GO TO 2
 1    CONTINUE
 2    IF(KEY.NE.1)GO TO 3
      I0=II-1
      I1=II
      GO TO 5
 3    IF(II.EQ.5)GO TO 4
      I0=II-1
      I1=II
      I2=II+1
      GO TO 5
 4    I0=II-2
      I1=II-1
      I2=II
C
C     SELECT THE BT11 INDICIES
C
 5    DO 6 J=2,6
      JJ=J
      IF(T.LE.TTAB(J))GO TO 7
 6    CONTINUE
 7    IF(KEY.NE.1)GO TO 8
      J0=JJ-1
      J1=JJ
      GO TO 10
 8    IF(JJ.EQ.6)GO TO 9
      J0=JJ-1
      J1=JJ
      J2=JJ+1
      GO TO 10
 9    J0=JJ-2
      J1=JJ-1
      J2=JJ
C
C     SET PARAMETER INDICIES
C
 10   continue
C
C     BRANCH ON SCHEME TYPE
C
      IF(KEY.NE.1)GO TO 20
C
C     LINEAR SCHEME
C
C     DESIGNATE INDEX VALUES
C
      U0=UTAB(I0)
      U1=UTAB(I1)
      T0=TTAB(J0)
      T1=TTAB(J1)
C
C     LAGRANGE POLYNOMIALS
C
      LU0=(U-U1)/(U0-U1)
      LU1=(U-U0)/(U1-U0)
      LT0=(T-T1)/(T0-T1)
      LT1=(T-T0)/(T1-T0)
C
C     LOOP OVER THE SCATTERING PARAMETER INDEX
C
C
C     INTERPOLATING POLYNOMIALS FOR THE FIRST DIMENSION
C
      P0=TAB(I0,J0)*LU0+TAB(I1,J0)*LU1
      P1=TAB(I0,J1)*LU0+TAB(I1,J1)*LU1
C
C     INTERPOLATING POLYNOMIAL FOR SECOND DIMENSION
C
      P=P0*LT0+P1*LT1
C
C     ASSIGN THE INDIVIDUAL PARAMETERS
C
      corr=p
 11   CONTINUE
      RETURN
C
C     QUADRATIC SCHEME
C
C     DESIGNATE INDEX VALUES
C
 20   U0=UTAB(I0)
      U1=UTAB(I1)
```

```
      U2=UTAB(I2)
      T0=TTAB(J0)
      T1=TTAB(J1)
      T2=TTAB(J2)
C
C     LAGRANGE POLYNOMIALS
C
      LU0=(U-U1)*(U-U2)/(U0-U1)/(U0-U2)
      LU1=(U-U0)*(U-U2)/(U1-U0)/(U1-U2)
      LU2=(U-U0)*(U-U1)/(U2-U0)/(U2-U1)
      LT0=(T-T1)*(T-T2)/(T0-T1)/(T0-T2)
      LT1=(T-T0)*(T-T2)/(T1-T0)/(T1-T2)
      LT2=(T-T0)*(T-T1)/(T2-T0)/(T2-T1)
C
C
C
C     INTERPOLATING POLYNOMIALS FOR THE FIRST DIMENSION
C
      P0=TAB(I0,J0)*LU0+TAB(I1,J0)*LU1+TAB(I2,J0)*LU2
      P1=TAB(I0,J1)*LU0+TAB(I1,J1)*LU1+TAB(I2,J1)*LU2
      P2=TAB(I0,J2)*LU0+TAB(I1,J2)*LU1+TAB(I2,J2)*LU2
C
C     INTERPOLATING POLYNOMIAL FOR SECOND DIMENSION
C
      P=P0*LT0+P1*LT1+P2*LT2
C
C     ASSIGN THE INDIVIDUAL PARAMETERS
C
      corr=P
 21   CONTINUE
      RETURN
      END
```

```
      REAL FUNCTION SOLR_IRRAD(WAVLEN,IDAY)

C     PURPOSE          CALCULATE SOLAR SPECTRAL IRRADIANCE INCIDENT AT
C                      TOP OF EARTH'S ATMOSPHERE AT A GIVEN WAVELENGTH.
C                      ALLOWANCE IS MADE FOR THE ELLIPTICAL ORBIT
C                      OF THE EARTH ABOUT THE SUN.
C                      BASED ON LOWTRAN7 SUBROUTINE "SOURCE"
C     INPUTS
C     WAVLEN           DESIRED WAVELENGTH IN MICRONS
C     DAY              DAY OF YEAR
C     OUTPUT
C     SOLAR            SOLAR SPECTRAL IRRADIANCE (MILLIWATTS/CM**2/MICR
C----------------------------------------------------------------------

      DIMENSION NDAY(13),RAT(13)
      DATA NDAY/1,32,60,91,121,152,182,213,244,274,305,335,366/
      DATA RAT/1.034,1.030,1.019,1.001,.985,.972,.967,.971,.982,
     1    .998,1.015,1.029,1.034/
CHANGE*** 2 AUG 89 LG
c     WRITE(*,*)'ENTERING SOLAR ',wavlen, iday
      VV=10000.0/WAVLEN
      V=10000./VV
      IF(VV.LE.0.) V = 1.0E+38
C
C     SUN ELLIPTIC ORBIT FACTOR
C
      FORBIT=0.0
      IF(IDAY.GT.0 .AND. IDAY.LT.367) GO TO 55
      FORBIT = 1.0
      GO TO 90
55    CONTINUE
      DO 60 I=1,13
      IF(NDAY(I).EQ.IDAY) GO TO 80
      IF(NDAY(I).GT.IDAY) GO TO 70
60    CONTINUE
70    FORBIT=RAT(I-1)+(IDAY-NDAY(I-1))*(RAT(I)-RAT(I-1))/(NDAY(I)
     1   -NDAY(I-1))
      GO TO 90
80    FORBIT=RAT(I)
90    CONTINUE
C
C     SOLAR INTENSITY
C
      SS = SUN(VV)*FORBIT
C     CONVERT W/M-2-MICRON TO MW/CM-2-MICRON
C     SOLR_IRRAD = SS*.1
      SOLR_IRRAD = SS
C
      RETURN
      END
C
      FUNCTION   SUN(V)
C
C        EVALUATES THE EXTRA-TERRESTRIAL SOLAR IRRADIANCE
C
C        INPUT:  V  =  FREQUENCY   (CM-1)
C                VALID RANGE   0 TO 57490 (CM-1)
C                   (EQUIVALENT TO WAVELENGTHS > 0.174 MICROMETERS)
C
C        OUTPUT:  SUN   =  SOLAR IRRADIANCE   (WATTS M-2 MICROMETER-1)
C
C        WRITES A WARNING MESSAGE TO TAPE6  &  RETURNS  SUN = 0
C            IF THE INPUT FREQUENCY IS OUT OF RANGE
C
C       USES  BLOCK DATA SOLAR   WHICH CONTAINS THE VALUES FOR SOLARA +
C
      COMMON /SUNDAT/ SOLARA(1440), SOLARB(2910)
      DATA  A, B / 3.50187E-13, 3.93281 /
C
C       WM, W0, W1, W2  ARE STATEMENT FUNKTIONS USED BY
C            THE 4 POINT LAGRANGE INTERPOLATION
      WM(P) = P*(P - 1)*(P - 2)
      W0(P) = 3*(P**2 - 1)*(P - 2)
      W1(P) = 3*P*(P + 1)*(P - 2)
      W2(P) = P*(P**2 - 1)
C
C           IF  V  IS TOO SMALL,  WRITE WARNING  +  RETURN SUN = 0
      IF(V .LT. 0.0) THEN
        SUN = 0.0
C       WRITE(6, 900)'MAS WAVENUMBER LESS THAN ZERO ',V
        RETURN
C
      ELSEIF( V .GE. 0.0   .AND.   V .LT. 100.0 ) THEN
C         FOR LOW FREQUENCIES USE A POWER LAW APPROXIMATION
        SUN = A*V**B
        RETURN
C
      ELSEIF( V .GE. 100.0   .AND.   V .LT. 28420.0 ) THEN
C         USE   4 POINT INTERPOLATION   ON   ARRAY   SOLARA
C               WHICH IS AT   20 CM-1   SPACING   FROM 0 TO 28720 CM-1
        I = 1 + INT(V/20.0)
        P = MOD(V, 20.0)/20.0
        SUN = ( W2(P)*SOLARA(I+2) - W1(P)*SOLARA(I+1) +
     +            W0(P)*SOLARA(I) - WM(P)*SOLARA(I-1) ) / 6
        RETURN
C
      ELSEIF( V .GE. 28420.0   .AND.   V .LE. 57470.0 ) THEN
C         USE   4 POINT INTERPOLATION   ON   ARRAY   SOLARB
C             WHICH IS AT   10 CM-1   SPACING   FROM 28400 TO 57490 CM-1
        I = INT(V/10.0) - 2839
        P = MOD(V, 10.0)/10.0
        SUN = ( W2(P)*SOLARB(I+2) - W1(P)*SOLARB(I+1) +
     +            W0(P)*SOLARB(I) - WM(P)*SOLARB(I-1) ) / 6
        RETURN
C
      ELSEIF( V .GT. 57470.0 ) THEN
C           IF  V  IS TOO LARGE,  WRITE WARNING  +  RETURN SUN = 0
        SUN = 0.0
        WRITE(6, 900)'MAS WAVENUMBER TOO LARGE ',V
        RETURN
C
      ENDIF
C
      RETURN
  900 FORMAT('0 *****  WARNING - INPUT FREQUENCY = ', 1PG12.5, 'CM-1',
     +  /, '   OUTSIDE VALID RANGE OF 0 TO 57470 CM-1    *******', / )
      END
      BLOCK DATA SOLARS
C>    BLOCK DATA
C
C
C     COMMON /SUNDAT/   SOLARA(1440), SOLARB(2910)
      COMMON /SUNDAT/   SUNA01( 41),SUNA02(144),SUNA03(144),SUNA04(144),
     A      SUNA05(144),SUNA06(144),SUNA07(144),SUNA08(144),SUNA09(144),
     B      SUNA10(144),SUNA11(103),SUNB01(144),SUNB02(144),SUNB03(144),
     C      SUNB04(144),SUNB05(144),SUNB06(144),SUNB07(144),SUNB08(144),
     D      SUNB09(144),SUNB10(144),SUNB11(144),SUNB12(144),SUNB13(144),
     E      SUNB14(144),SUNB15(144),SUNB16(144),SUNB17(144),SUNB18(144),
     F      SUNB19(144),SUNB20(144),SUNB21( 30)
C
```

```
C         SOLAR SPECTRUM FROM      0 TO    800 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA01  /
    A 0.0000E+00, 4.5756E-08, 7.0100E-07, 3.4580E-06, 1.0728E-05,
    B 2.5700E-05, 5.2496E-05, 9.6003E-05, 1.6193E-04, 2.5766E-04,
    C 3.9100E-04, 5.6923E-04, 8.0203E-04, 1.1006E-03, 1.4768E-03,
    D 1.9460E-03, 2.5213E-03, 3.2155E-03, 4.0438E-03, 5.0229E-03,
    E 6.1700E-03, 7.5145E-03, 9.0684E-03, 1.0853E-02, 1.2889E-02,
    F 1.5213E-02, 1.7762E-02, 2.0636E-02, 2.3888E-02, 2.7524E-02,
    G 3.1539E-02, 3.5963E-02, 4.0852E-02, 4.6236E-02, 5.2126E-02,
    H 5.8537E-02, 6.5490E-02, 7.3017E-02, 8.1169E-02, 9.0001E-02,
    I 9.9540E-02 /
C         SOLAR SPECTRUM FROM    820 TO   3680 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA02  /
    A .10980, .12080, .13260, .14520, .15860, .17310, .18850, .20490,
    B .22240, .24110, .26090, .28200, .30430, .32790, .35270, .37890,
    C .40650, .43550, .46600, .49800, .53160, .56690, .60390, .64260,
    D .68320, .72560, .76990, .81620, .86440, .91470, .96710, 1.0220,
    E 1.0780, 1.1370, 1.1990, 1.2630, 1.3290, 1.3990, 1.4710, 1.5460,
    F 1.6250, 1.7060, 1.7910, 1.8800, 1.9710, 2.0670, 2.1660, 2.2680,
    G 2.3740, 2.4840, 2.5970, 2.7140, 2.8350, 2.9600, 3.0890, 3.2210,
    H 3.3570, 3.4980, 3.6420, 3.7900, 3.9440, 4.1040, 4.2730, 4.4450,
    I 4.6150, 4.7910, 4.9830, 5.1950, 5.4210, 5.6560, 5.8930, 6.1270,
    J 6.3560, 6.5820, 6.8080, 7.0360, 7.2700, 7.5170, 7.7890, 8.0910,
    K 8.4070, 8.7120, 8.9900, 9.2490, 9.5000, 9.7550, 10.010, 10.250,
    L 10.480, 10.700, 10.950, 11.230, 11.550, 11.900, 12.250, 12.600,
    M 12.930, 13.250, 13.530, 13.780, 14.040, 14.320, 14.660, 15.070,
    N 15.530, 16.011, 16.433, 16.771, 17.077, 17.473, 17.964, 18.428,
    O 18.726, 18.906, 19.141, 19.485, 19.837, 20.160, 20.509, 21.024,
    P 21.766, 22.568, 23.190, 23.577, 23.904, 24.335, 24.826, 25.236,
    Q 25.650, 26.312, 27.208, 27.980, 28.418, 28.818, 29.565, 30.533,
    R 31.247, 31.667, 32.221, 33.089, 33.975, 34.597, 35.004, 35.395 /
C         SOLAR SPECTRUM FROM   3700 TO   6560 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA03  /
    A 36.026, 36.985, 37.890, 38.401, 38.894, 39.857, 40.926, 41.570,
    B 42.135, 43.083, 44.352, 45.520, 45.982, 46.281, 48.335, 51.987,
    C 54.367, 54.076, 52.174, 50.708, 52.153, 55.707, 56.549, 54.406,
    D 53.267, 56.084, 61.974, 64.406, 60.648, 55.146, 53.067, 57.476,
    E 64.645, 68.348, 69.055, 69.869, 70.943, 71.662, 72.769, 74.326,
    F 75.257, 74.883, 73.610, 73.210, 74.886, 78.042, 80.204, 80.876,
    G 82.668, 84.978, 86.244, 88.361, 91.998, 95.383, 98.121, 100.29,
    H 100.64, 99.997, 101.82, 105.06, 107.50, 109.99, 112.45, 113.90,
    I 113.79, 119.23, 121.96, 124.58, 127.14, 125.19, 124.37, 125.00,
    J 127.88, 130.67, 131.98, 133.74, 136.69, 136.18, 135.02, 137.44,
    K 138.44, 137.25, 136.35, 142.60, 144.54, 148.37, 151.90, 151.55,
    L 155.35, 157.59, 159.70, 162.28, 168.44, 171.43, 169.82, 170.33,
    M 172.28, 176.68, 181.92, 186.06, 187.85, 186.00, 189.82, 189.35,
    N 192.86, 202.00, 209.63, 205.76, 212.88, 215.63, 216.51, 219.20,
    O 220.29, 221.12, 227.12, 229.97, 233.23, 233.95, 234.52, 234.45,
    P 235.77, 239.80, 243.11, 241.19, 242.34, 243.69, 242.84, 246.19,
    Q 246.11, 246.76, 251.75, 255.38, 258.74, 260.26, 263.40, 268.68,
    R 271.81, 272.95, 273.93, 274.74, 274.43, 279.69, 287.76, 287.72 /
C         SOLAR SPECTRUM FROM   6580 TO   9440 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA04  /
    A 287.96, 290.01, 291.92, 295.28, 296.78, 300.46, 302.19, 299.14,
    B 301.43, 305.68, 309.29, 310.63, 313.24, 314.61, 309.58, 318.81,
    C 320.54, 321.62, 328.58, 331.66, 337.20, 345.62, 345.54, 342.96,
    D 344.38, 346.23, 349.17, 351.79, 354.71, 356.97, 358.29, 362.29,
    E 364.15, 364.97, 367.81, 368.98, 369.07, 372.17, 377.79, 381.25,
    F 384.22, 388.66, 393.58, 396.98, 398.72, 400.61, 404.06, 408.23,
    G 412.47, 415.58, 416.17, 416.53, 419.55, 425.88, 433.30, 437.73,
    H 438.13, 439.79, 441.51, 438.71, 434.25, 437.54, 448.95, 448.86,
    I 439.46, 437.10, 439.34, 444.33, 455.00, 467.05, 473.04, 469.64,
    J 467.53, 473.78, 477.50, 477.50, 480.96, 483.94, 482.19, 479.08,
    K 482.09, 493.43, 498.40, 492.05, 489.53, 493.34, 495.51, 496.52,
    L 499.57, 504.65, 509.68, 512.00, 512.05, 512.31, 515.00, 520.70,
    M 527.30, 531.88, 532.16, 530.48, 532.33, 539.26, 548.57, 553.00,
    N 548.96, 546.05, 551.00, 556.41, 557.21, 557.85, 560.95, 564.02,
    O 565.57, 566.38, 567.88, 571.48, 576.68, 581.54, 586.51, 593.62,
    P 600.70, 602.79, 601.39, 603.00, 606.88, 605.95, 600.97, 600.79,
    Q 607.21, 612.87, 614.13, 614.39, 616.61, 620.53, 625.19, 629.78,
    R 633.79, 637.31, 640.47, 642.53, 642.62, 641.93, 643.11, 646.68 /
C         SOLAR SPECTRUM FROM   9460 TO  12320 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA05  /
    A 650.57, 654.30, 660.95, 672.10, 682.31, 684.89, 682.20, 682.53,
    B 687.79, 691.42, 689.62, 688.14, 693.71, 703.25, 708.07, 706.22,
    C 704.64, 708.97, 717.35, 725.43, 731.08, 734.17, 735.41, 736.60,
    D 739.34, 742.90, 745.04, 744.29, 742.44, 749.53, 755.70, 758.82,
    E 766.31, 761.53, 762.09, 769.68, 764.18, 763.75, 768.88, 762.69,
    F 753.93, 762.38, 765.79, 772.19, 760.67, 762.10, 766.76, 766.98,
    G 769.35, 773.50, 766.84, 763.60, 773.82, 777.18, 779.61, 792.48,
    H 797.54, 787.81, 793.75, 805.96, 804.77, 806.62, 821.72, 830.28,
    I 827.54, 831.06, 830.20, 826.22, 823.28, 822.18, 833.92, 854.58,
    J 859.80, 862.56, 871.16, 875.16, 867.67, 863.87, 883.30, 893.40,
    K 897.74, 905.24, 905.38, 911.07, 930.21, 939.24, 934.74, 935.15,
    L 942.38, 948.13, 947.00, 951.88, 960.12, 951.88, 954.22, 959.07,
    M 963.36, 980.16, 983.66, 978.76, 979.38, 985.24, 977.08, 919.94,
    N 899.68, 962.91, 997.17, 999.93, 995.65, 999.93, 1014.9, 951.57,
    O 893.52, 955.14, 1003.1, 990.13, 978.79, 1011.2, 1034.7, 1031.9,
    P 1029.9, 1039.7, 1045.5, 1044.1, 1049.6, 1056.1, 1049.8, 1038.0,
    Q 1051.9, 1072.2, 1075.5, 1077.0, 1079.3, 1078.0, 1075.7, 1079.7,
    R 1081.0, 1069.8, 1078.4, 1104.3, 1111.4, 1111.7, 1117.6, 1119.6 /
C         SOLAR SPECTRUM FROM  12340 TO  15200 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA06  /
    A 1109.3, 1100.6, 1112.9, 1122.7, 1119.5, 1123.9, 1136.1, 1143.7,
    B 1140.5, 1141.2, 1151.5, 1148.7, 1138.3, 1141.0, 1150.6, 1160.1,
    C 1170.6, 1177.7, 1179.8, 1181.7, 1182.4, 1179.8, 1181.8, 1188.3,
    D 1190.0, 1191.4, 1197.0, 1196.0, 1192.2, 1200.6, 1210.4, 1209.1,
    E 1207.5, 1205.3, 1193.3, 1192.9, 1220.0, 1243.3, 1245.4, 1241.5,
    F 1240.2, 1241.1, 1244.0, 1248.5, 1253.2, 1257.1, 1259.9, 1261.9,
    G 1263.6, 1265.7, 1269.6, 1277.0, 1284.2, 1284.4, 1282.7, 1287.2,
    H 1286.8, 1272.3, 1262.2, 1270.7, 1288.8, 1304.8, 1311.8, 1312.2,
    I 1314.4, 1320.2, 1326.2, 1328.4, 1325.3, 1322.5, 1325.4, 1334.6,
    J 1346.4, 1354.0, 1353.7, 1347.3, 1338.3, 1331.0, 1329.7, 1338.0,
    K 1351.9, 1363.0, 1368.8, 1372.0, 1375.9, 1382.1, 1387.8, 1388.8,
    L 1388.2, 1392.2, 1401.7, 1412.9, 1418.2, 1410.7, 1395.9, 1385.7,
    M 1388.1, 1405.0, 1424.0, 1428.1, 1422.2, 1423.6, 1434.5, 1445.2,
    N 1450.7, 1451.8, 1451.5, 1453.9, 1459.9, 1466.9, 1471.3, 1469.4,
    O 1462.5, 1460.4, 1468.9, 1481.8, 1490.8, 1495.3, 1497.9, 1500.7,
    P 1505.2, 1510.0, 1512.3, 1512.7, 1515.6, 1521.6, 1524.2, 1520.7,
    Q 1520.3, 1531.6, 1545.7, 1548.2, 1541.7, 1542.2, 1553.6, 1563.6,
    R 1563.6, 1559.9, 1561.3, 1569.9, 1581.6, 1577.6, 1529.7, 1447.0 /
C         SOLAR SPECTRUM FROM  15220 TO  18080 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA07  /
    A 1396.9, 1428.7, 1506.4, 1567.1, 1594.0, 1606.1, 1613.5, 1609.0,
    B 1588.6, 1567.8, 1567.3, 1587.2, 1610.2, 1624.4, 1630.2, 1630.9,
    C 1628.1, 1622.3, 1616.9, 1618.9, 1631.6, 1648.1, 1658.2, 1659.7,
    D 1658.1, 1658.0, 1659.4, 1660.4, 1659.2, 1653.7, 1645.3, 1642.1,
    E 1652.7, 1674.2, 1694.1, 1700.6, 1703.4, 1697.6, 1654.5, 1644.4,
    F 1661.6, 1676.3, 1707.7, 1703.1, 1710.8, 1732.3, 1716.5, 1719.6,
    G 1729.6, 1683.1, 1628.5, 1683.5, 1727.0, 1707.8, 1689.4, 1698.4,
    H 1733.1, 1737.8, 1714.1, 1734.6, 1750.1, 1750.1, 1760.3, 1764.3,
    I 1765.3, 1769.4, 1779.9, 1793.0, 1765.1, 1729.4, 1745.9, 1753.4,
    J 1758.1, 1775.0, 1768.4, 1767.9, 1789.5, 1806.6, 1799.3, 1782.6,
    K 1779.3, 1792.1, 1809.7, 1808.0, 1794.4, 1818.6, 1774.2, 1648.5,
    L 1674.3, 1789.3, 1847.2, 1848.3, 1812.9, 1796.4, 1840.3, 1868.3,
    M 1864.6, 1873.2, 1872.2, 1856.0, 1845.0, 1842.4, 1823.9, 1795.1,
    N 1819.6, 1861.5, 1857.7, 1838.6, 1840.5, 1863.5, 1876.8, 1884.4,
    O 1894.9, 1875.2, 1821.2, 1779.4, 1810.2, 1855.3, 1831.8, 1837.3,
    P 1882.3, 1866.4, 1819.6, 1804.8, 1831.4, 1861.6, 1867.1, 1862.9,
    Q 1851.9, 1834.7, 1835.2, 1845.1, 1831.9, 1803.6, 1792.5, 1821.8,
```

```
     R 1845.8, 1832.3, 1847.6, 1894.2, 1909.2, 1901.0, 1891.2, 1869.9 /
C        SOLAR SPECTRUM FROM  18100 TO  20960 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA08  /
     A 1854.4, 1865.8, 1873.7, 1868.8, 1881.7, 1897.1, 1884.2, 1856.2,
     B 1840.6, 1855.1, 1885.3, 1903.6, 1900.1, 1887.4, 1887.7, 1879.0,
     C 1844.5, 1844.1, 1877.1, 1847.3, 1785.1, 1792.6, 1848.7, 1894.4,
     D 1908.8, 1892.8, 1867.4, 1885.6, 1959.9, 1971.9, 1895.8, 1883.5,
     E 1917.6, 1853.8, 1793.0, 1875.6, 1974.0, 1975.7, 1943.9, 1926.4,
     F 1914.4, 1902.7, 1882.5, 1813.3, 1710.8, 1717.9, 1859.7, 1965.1,
     G 1970.1, 1941.4, 1902.5, 1852.0, 1836.3, 1879.3, 1901.6, 1862.9,
     H 1839.1, 1840.9, 1780.0, 1684.9, 1677.3, 1718.7, 1697.3, 1684.3,
     I 1784.5, 1898.0, 1910.3, 1877.2, 1866.6, 1862.6, 1860.3, 1899.7,
     J 1971.0, 1999.9, 1970.9, 1936.5, 1922.8, 1922.8, 1924.0, 1917.2,
     K 1912.0, 1926.2, 1959.7, 1995.4, 1995.9, 1938.8, 1883.5, 1894.7,
     L 1933.3, 1935.1, 1899.3, 1852.7, 1820.2, 1821.5, 1865.2, 1935.5,
     M 1966.1, 1919.6, 1881.2, 1931.5, 2015.6, 2050.0, 2021.4, 1960.8,
     N 1938.2, 1997.0, 2051.0, 2003.4, 1912.1, 1880.2, 1895.2, 1898.0,
     O 1898.8, 1938.3, 1994.2, 2010.0, 1982.4, 1948.8, 1927.3, 1911.6,
     P 1877.7, 1791.6, 1679.8, 1645.0, 1727.3, 1845.2, 1926.2, 1973.4,
     Q 2005.2, 2021.6, 2021.8, 2025.7, 2054.3, 2086.5, 2082.6, 2052.9,
     R 2047.1, 2070.2, 2072.4, 2038.1, 2020.2, 2049.9, 2074.0, 2038.1 /
C        SOLAR SPECTRUM FROM  20980 TO  23840 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA09  /
     A 1978.6, 1963.5, 1996.8, 2037.5, 2057.5, 2048.2, 2018.4, 1999.2,
     B 2011.4, 2039.5, 2056.0, 2040.2, 1981.8, 1911.4, 1891.8, 1938.3,
     C 1991.7, 2005.5, 2000.8, 2011.3, 2022.7, 1997.5, 1947.7, 1936.3,
     D 1986.6, 2037.9, 2032.8, 1995.7, 1984.0, 2012.0, 2055.5, 2091.6,
     E 2106.5, 2094.9, 2070.4, 2052.8, 2046.7, 2043.8, 2035.5, 2016.6,
     F 1988.4, 1973.3, 1999.0, 2057.4, 2103.8, 2109.4, 2089.4, 2068.5,
     G 2051.8, 2031.2, 2005.9, 1986.7, 1981.5, 1979.4, 1964.1, 1943.6,
     H 1951.8, 2007.3, 2083.2, 2139.1, 2158.0, 2143.3, 2103.2, 2050.9,
     I 2001.9, 1974.5, 1988.0, 2037.8, 2075.1, 2050.6, 1971.5, 1884.5,
     J 1828.5, 1820.9, 1866.4, 1935.3, 1974.2, 1958.7, 1925.1, 1920.2,
     K 1949.7, 1984.6, 1996.4, 1966.4, 1884.8, 1781.9, 1726.8, 1759.4,
     L 1817.4, 1800.4, 1692.6, 1593.2, 1598.6, 1700.3, 1823.8, 1909.7,
     M 1937.7, 1902.5, 1822.4, 1737.8, 1683.2, 1666.8, 1682.7, 1715.3,
     N 1734.1, 1712.4, 1668.2, 1655.0, 1698.1, 1727.2, 1636.9, 1415.7,
     O 1204.2, 1155.8, 1278.4, 1450.0, 1560.5, 1595.1, 1587.8, 1570.6,
     P 1565.8, 1590.3, 1640.5, 1688.4, 1708.1, 1703.6, 1700.7, 1718.5,
     Q 1749.0, 1772.2, 1772.5, 1745.2, 1690.2, 1624.9, 1589.0, 1618.5,
     R 1701.3, 1783.2, 1816.4, 1800.7, 1765.0, 1734.1, 1714.6, 1705.0 /
C        SOLAR SPECTRUM FROM  23860 TO  26720 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA10  /
     A 1701.6, 1696.6, 1682.0, 1661.4, 1657.2, 1693.0, 1763.2, 1826.5,
     B 1841.6, 1806.1, 1755.6, 1725.8, 1724.2, 1736.8, 1749.0, 1756.1,
     C 1759.5, 1762.1, 1770.2, 1791.7, 1826.8, 1848.9, 1819.6, 1720.7,
     D 1595.5, 1513.9, 1522.5, 1602.0, 1706.2, 1793.4, 1837.9, 1820.3,
     E 1738.3, 1631.1, 1553.1, 1539.2, 1574.3, 1623.9, 1660.6, 1676.8,
     F 1673.1, 1652.9, 1626.4, 1606.7, 1604.2, 1620.9, 1654.5, 1701.2,
     G 1752.2, 1796.2, 1822.8, 1827.4, 1808.5, 1767.0, 1713.9, 1667.3,
     H 1643.7, 1643.5, 1652.5, 1655.3, 1638.7, 1592.2, 1506.4, 1377.3,
     I 1209.5, 1010.5, 807.59, 666.84, 664.53, 835.23, 1099.6, 1330.7,
     J 1423.2, 1363.7, 1194.1, 961.77, 725.04, 551.29, 504.01, 596.30,
     K 775.15, 975.62, 1150.2, 1287.2, 1386.1, 1447.5, 1473.7, 1468.5,
     L 1435.2, 1376.9, 1296.0, 1195.5, 1085.3, 985.40, 917.25, 894.59,
     M 910.86, 951.53, 1001.7, 1046.4, 1070.7, 1061.2, 1021.2, 977.16,
     N 959.15, 982.06, 1020.5, 1032.6, 983.44, 879.83, 762.66, 675.28,
     O 643.33, 662.65, 721.49, 808.35, 913.24, 1027.0, 1139.9, 1236.2,
     P 1293.2, 1287.1, 1210.4, 1102.1, 1021.6, 1022.8, 1109.3, 1232.6,
     Q 1337.0, 1383.1, 1372.8, 1324.7, 1257.7, 1188.8, 1133.5, 1106.5,
     R 1113.7, 1136.8, 1147.9, 1121.4, 1054.1, 968.10, 889.19, 837.87 /
C        SOLAR SPECTRUM FROM  26740 TO  28780 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA11  /
     A 817.64, 823.72, 851.04, 896.53, 959.85, 1041.2, 1137.6, 1231.2,
     B 1294.4, 1299.9, 1241.2, 1155.0, 1092.0, 1097.1, 1170.2, 1263.5,
```

```
     C 1322.4, 1307.4, 1233.6, 1146.1, 1090.8, 1092.5, 1134.6, 1188.9,
     D 1228.9, 1245.5, 1248.5, 1250.3, 1260.5, 1274.6, 1279.5, 1261.8,
     E 1214.3, 1145.4, 1069.6, 1001.4, 952.52, 930.48, 941.68, 990.34,
     F 1064.4, 1135.2, 1171.5, 1149.1, 1076.3, 984.35, 906.25, 868.17,
     G 873.75, 915.33, 984.41, 1067.2, 1137.1, 1163.1, 1115.5, 990.55,
     H 830.93, 692.29, 627.44, 654.10, 739.24, 838.88, 911.69, 941.90,
     I 944.42, 939.58, 946.10, 970.23, 1005.2, 1042.4, 1073.8, 1097.0,
     J 1114.3, 1128.8, 1142.9, 1153.4, 1152.4, 1131.5, 1084.2, 1016.7,
     K 945.95, 890.37, 866.15, 876.54, 913.13, 966.10, 1025.4, 1080.2,
     L 1119.0, 1102.7, 1243.5, 1209.9, 1079.2, 852.20, 956.80, 842.31,
     M 897.44, 1081.8, 914.23, 993.09, 1049.8, 844.95, 839.16/
C        SOLAR SPECTRUM FROM  28400 TO  29830 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB01  /
     A 876.54, 892.17, 913.13, 938.18, 966.10, 995.62, 1025.4, 1054.1,
     B 1080.2, 1102.1, 1119.0, 1132.2, 1102.7, 1159.3, 1243.5, 1238.3,
     C 1209.9, 1196.2, 1079.2, 895.60, 852.20, 935.59, 956.80, 897.09,
     D 842.31, 821.15, 897.44, 1042.7, 1081.8, 988.79, 914.23, 929.38,
     E 993.09, 1041.9, 1049.8, 984.33, 844.95, 770.76, 839.16, 939.65,
     F 1026.1, 1121.1, 1162.6, 1142.6, 1077.9, 1027.3, 1078.2, 1094.3,
     G 969.83, 853.72, 849.91, 909.12, 995.68, 1095.0, 1146.9, 1086.3,
     H 1010.4, 1065.4, 1128.9, 1080.6, 987.93, 898.18, 835.20, 771.63,
     I 687.12, 614.52, 606.14, 737.09, 908.13, 997.64, 1080.6, 1126.3,
     J 1056.7, 1028.4, 1141.7, 1252.6, 1225.3, 1103.2, 1038.6, 1043.4,
     K 1002.9, 965.51, 1035.0, 1150.7, 1200.9, 1152.0, 1068.5, 995.84,
     L 889.52, 818.48, 907.01, 1042.2, 1055.6, 1000.6, 972.00, 985.72,
     M 1027.2, 1054.8, 1078.0, 1126.6, 1205.3, 1245.7, 1201.0, 1144.7,
     N 1097.5, 1030.1, 926.85, 836.71, 864.11, 993.50, 1075.3, 1032.6,
     O 1008.9, 1066.1, 1067.4, 1004.8, 971.54, 923.18, 815.71, 799.70,
     P 946.19, 1100.1, 1126.4, 1032.2, 895.14, 784.30, 734.77, 726.53,
     Q 726.88, 765.54, 863.90, 992.24, 1070.9, 1028.1, 858.78, 647.15,
     R 563.18, 679.98, 906.40, 1094.3, 1155.3, 1124.3, 1098.4, 1109.5 /
C        SOLAR SPECTRUM FROM  29840 TO  31270 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB02  /
     A 1076.2, 944.17, 849.20, 928.54, 1062.0, 1118.9, 1119.2, 1074.6,
     B 1005.8, 980.02, 999.11, 1002.4, 939.78, 838.12, 816.13, 908.73,
     C 1014.9, 1058.3, 1043.7, 987.54, 946.35, 981.40, 1055.8, 1094.3,
     D 1028.3, 916.41, 908.99, 991.83, 1049.6, 1076.2, 1093.5, 1076.3,
     E 1014.5, 949.61, 947.26, 1001.2, 1051.5, 1072.8, 1068.0, 1012.5,
     F 907.81, 866.30, 950.89, 1037.5, 1079.5, 1183.9, 1291.3, 1268.6,
     G 1199.3, 1188.6, 1188.0, 1186.6, 1198.2, 1171.3, 1132.6, 1131.6,
     H 1096.0, 971.10, 847.07, 836.62, 922.78, 990.99, 987.51, 969.24,
     I 981.46, 981.36, 971.95, 985.34, 1003.0, 1037.2, 1071.2, 1065.7,
     J 1026.7, 984.84, 1002.7, 1070.3, 1117.5, 1116.0, 1048.9, 965.34,
     K 972.27, 1045.7, 1096.6, 1127.5, 1133.5, 1099.6, 1079.3, 1082.9,
     L 1026.8, 927.50, 879.08, 858.83, 831.01, 807.82, 789.56, 813.75,
     M 893.46, 937.62, 901.56, 864.46, 873.35, 891.03, 862.46, 810.30,
     N 787.36, 752.93, 715.34, 708.07, 728.93, 786.79, 807.73, 736.28,
     O 645.08, 616.90, 649.17, 691.77, 749.18, 820.21, 820.68, 791.26,
     P 854.27, 940.56, 956.38, 909.42, 824.18, 767.17, 722.06, 653.42,
     Q 624.67, 633.73, 655.14, 707.93, 784.94, 880.79, 961.15, 985.60,
     R 986.18, 966.53, 921.47, 888.89, 855.85, 851.66, 886.78, 850.97 /
C        SOLAR SPECTRUM FROM  31280 TO  32710 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB03  /
     A 766.97, 738.95, 724.53, 657.61, 587.77, 616.86, 760.61, 903.23,
     B 917.27, 838.49, 784.80, 759.41, 719.61, 671.48, 624.63, 588.57,
     C 574.70, 596.68, 698.02, 866.39, 974.82, 960.37, 930.10, 962.65,
     D 1007.1, 1001.9, 926.29, 816.64, 763.25, 772.93, 762.66, 729.39,
     E 725.01, 727.16, 672.73, 581.42, 520.97, 488.80, 478.60, 542.08,
     F 663.71, 749.48, 785.87, 811.05, 818.19, 813.80, 824.54, 836.62,
     G 799.66, 728.00, 660.36, 559.28, 473.28, 550.16, 752.04, 885.84,
     H 906.80, 912.21, 929.32, 899.72, 830.20, 774.56, 736.42, 724.09,
     I 740.12, 754.11, 764.96, 780.76, 788.94, 784.87, 758.80, 725.91,
     J 751.84, 804.24, 777.73, 703.36, 665.27, 663.99, 679.36, 706.09,
     K 757.57, 836.09, 880.02, 881.18, 907.91, 929.26, 894.32, 874.01,
     L 918.56, 953.50, 922.32, 866.61, 836.54, 825.28, 752.54, 586.02,
```

```
     M 427.46, 374.05, 437.23, 534.32, 556.74, 563.11, 629.31, 631.26,
     N 518.76, 438.31, 460.31, 530.45, 608.50, 657.99, 662.08, 686.17,
     O 775.18, 843.11, 797.46, 685.33, 611.33, 628.74, 711.36, 754.94,
     P 728.80, 722.79, 726.38, 679.68, 665.83, 710.48, 723.10, 724.09,
     Q 760.18, 784.01, 742.78, 634.33, 546.55, 563.54, 611.03, 623.16,
     R 665.36, 743.55, 764.46, 671.14, 513.18, 401.86, 405.77, 515.72 /
C        SOLAR SPECTRUM FROM  32720 TO  34150 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB04  /
     A 639.90, 677.85, 679.55, 759.33, 848.11, 819.89, 751.75, 710.50,
     B 615.33, 525.09, 583.35, 715.23, 767.53, 739.10, 664.05, 580.57,
     C 572.85, 634.13, 648.77, 561.27, 497.72, 591.71, 737.83, 794.19,
     D 802.51, 799.33, 735.79, 658.41, 659.47, 718.18, 761.67, 697.24,
     E 545.14, 474.47, 526.96, 597.65, 584.74, 447.28, 291.35, 261.28,
     F 330.26, 401.96, 466.32, 531.26, 572.34, 584.86, 585.17, 569.46,
     G 558.27, 559.41, 512.02, 426.37, 378.14, 398.26, 473.49, 542.18,
     H 531.76, 437.48, 341.85, 305.82, 299.88, 328.12, 440.04, 586.46,
     I 660.32, 625.22, 510.26, 418.85, 447.36, 534.89, 605.86, 667.07,
     J 687.31, 636.79, 549.63, 472.88, 419.53, 370.06, 327.98, 320.49,
     K 354.00, 399.17, 450.98, 528.34, 608.25, 696.07, 774.28, 760.75,
     L 690.58, 648.20, 580.63, 477.96, 453.91, 488.74, 464.02, 421.59,
     M 444.32, 446.59, 375.95, 342.13, 397.49, 510.97, 646.38, 725.14,
     N 703.06, 639.06, 619.10, 654.66, 665.99, 611.40, 580.22, 607.29,
     O 591.05, 542.30, 583.82, 673.02, 673.21, 582.44, 465.73, 377.25,
     P 377.04, 487.27, 607.93, 617.52, 583.46, 601.68, 615.94, 575.47,
     Q 541.63, 542.06, 522.28, 472.49, 423.29, 438.09, 556.72, 664.34,
     R 669.88, 657.45, 684.71, 705.70, 683.11, 600.81, 509.90, 497.64 /
C        SOLAR SPECTRUM FROM  34160 TO  35590 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB05  /
     A 511.07, 496.07, 500.32, 518.70, 529.91, 563.00, 609.20, 626.49,
     B 622.11, 615.72, 600.44, 591.26, 598.12, 593.07, 590.94, 631.58,
     C 696.48, 718.48, 676.11, 631.56, 619.64, 620.53, 624.10, 636.56,
     D 658.02, 688.78, 724.81, 742.60, 722.31, 675.86, 665.96, 704.73,
     E 703.70, 645.00, 598.26, 587.77, 590.94, 575.93, 528.03, 477.92,
     F 457.52, 456.80, 454.91, 448.65, 445.47, 445.38, 444.43, 446.04,
     G 455.91, 468.02, 454.34, 393.32, 301.22, 211.44, 167.11, 193.99,
     H 254.01, 305.35, 353.03, 385.08, 387.03, 391.60, 406.20, 415.34,
     I 435.34, 469.77, 492.15, 472.73, 409.86, 353.25, 340.68, 355.27,
     J 379.77, 401.81, 409.67, 406.89, 393.16, 378.89, 375.20, 373.52,
     K 360.19, 322.79, 273.55, 237.76, 212.33, 184.80, 156.20, 127.75,
     L 96.269, 68.806, 62.047, 77.143, 100.47, 127.56, 159.88, 194.05,
     M 225.20, 254.64, 285.75, 300.14, 294.40, 308.92, 340.83, 346.26,
     N 336.29, 347.54, 373.81, 388.78, 372.68, 325.29, 294.40, 317.56,
     O 360.30, 378.08, 374.22, 374.03, 383.34, 387.88, 377.55, 356.96,
     P 340.67, 328.71, 314.00, 316.91, 344.51, 355.54, 335.66, 318.68,
     Q 318.65, 322.43, 318.61, 304.92, 284.84, 268.13, 265.80, 273.55,
     R 274.18, 252.38, 215.04, 188.60, 181.31, 181.31, 180.78, 175.24 /
C        SOLAR SPECTRUM FROM  35600 TO  37030 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB06  /
     A 162.06, 145.08, 128.76, 113.76, 98.078, 83.072, 76.222, 78.359,
     B 78.434, 74.235, 75.843, 80.321, 77.859, 70.298, 64.651, 67.049,
     C 77.810, 83.167, 75.286, 71.202, 80.549, 92.008, 100.17, 108.63,
     D 119.44, 130.78, 142.31, 158.94, 177.12, 186.40, 186.60, 181.47,
     E 175.30, 175.54, 179.00, 177.04, 172.60, 172.67, 178.98, 193.77,
     F 215.13, 233.62, 252.05, 277.68, 298.91, 298.40, 280.81, 274.21,
     G 286.52, 285.46, 259.71, 241.39, 246.98, 259.87, 274.27, 298.47,
     H 316.85, 303.19, 263.69, 229.31, 227.90, 256.12, 281.58, 300.19,
     I 310.56, 279.54, 211.93, 152.18, 129.94, 147.47, 181.62, 215.37,
     J 239.50, 233.12, 191.55, 139.41, 110.51, 118.93, 134.79, 129.05,
     K 124.39, 143.53, 158.29, 141.84, 116.32, 111.59, 128.93, 149.17,
     L 153.44, 145.63, 148.52, 159.25, 155.84, 154.17, 177.28, 203.40,
     M 207.35, 205.27, 222.85, 253.18, 271.28, 279.27, 302.17, 321.47,
     N 288.83, 230.14, 206.40, 213.22, 216.49, 207.46, 196.20, 195.21,
     O 202.03, 194.33, 164.86, 136.65, 123.87, 128.14, 161.89, 216.99,
     P 253.68, 249.26, 222.89, 213.11, 243.64, 293.10, 309.42, 286.40,
     Q 269.61, 272.23, 271.67, 265.84, 265.61, 264.77, 266.03, 289.51,
     R 325.67, 337.34, 321.17, 300.30, 282.60, 287.14, 322.06, 335.79 /
C        SOLAR SPECTRUM FROM  37040 TO  38470 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB07  /
     A 297.22, 254.10, 243.47, 239.49, 219.32, 211.94, 239.28, 271.43,
     B 279.37, 272.26, 264.77, 250.52, 229.93, 222.15, 235.30, 256.79,
     C 275.28, 286.92, 284.85, 269.52, 255.05, 253.46, 263.22, 274.78,
     D 279.19, 270.17, 249.41, 229.04, 221.64, 231.38, 252.70, 280.64,
     E 310.06, 328.33, 325.01, 290.26, 238.97, 223.38, 257.24, 282.60,
     F 264.32, 243.34, 253.18, 272.89, 271.32, 256.12, 260.24, 271.35,
     G 257.11, 236.61, 238.72, 248.92, 255.90, 272.04, 291.78, 297.40,
     H 288.09, 283.28, 292.92, 301.74, 309.07, 322.05, 320.42, 295.43,
     I 269.65, 254.41, 240.88, 228.18, 221.23, 213.72, 201.23, 197.17,
     J 212.29, 233.39, 247.65, 261.74, 286.17, 322.49, 349.47, 338.28,
     K 297.06, 261.55, 252.28, 264.65, 286.92, 298.94, 280.45, 244.37,
     L 213.47, 193.03, 182.07, 168.54, 143.12, 114.10, 89.615, 73.589,
     M 73.990, 87.912, 96.265, 94.813, 96.604, 102.30, 102.15, 103.07,
     N 117.81, 137.41, 146.09, 144.28, 137.89, 128.11, 122.82, 128.19,
     O 130.66, 117.31, 98.912, 93.397, 105.63, 122.73, 126.39, 113.05,
     P 92.317, 76.340, 69.032, 66.324, 71.280, 87.431, 105.94, 114.02,
     Q 107.91, 91.872, 75.208, 69.123, 75.930, 90.928, 109.71, 125.70,
     R 135.79, 141.14, 138.14, 121.33, 91.806, 63.497, 52.106, 59.555 /
C        SOLAR SPECTRUM FROM  38480 TO  39910 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB08  /
     A 81.015, 106.67, 118.97, 116.36, 110.82, 100.88, 89.056, 90.431,
     B 104.41, 114.95, 124.85, 148.87, 171.72, 167.22, 142.25, 118.42,
     C 98.653, 78.908, 68.133, 77.286, 100.93, 120.08, 125.49, 131.79,
     D 155.69, 180.75, 181.81, 166.77, 150.06, 133.24, 116.14, 97.728,
     E 81.629, 76.695, 87.607, 110.23, 134.88, 149.13, 147.64, 139.88,
     F 135.19, 135.07, 138.00, 136.73, 128.84, 122.22, 120.48, 121.98,
     G 123.08, 116.30, 101.43, 86.303, 74.719, 68.800, 71.327, 80.626,
     H 90.485, 96.739, 100.69, 100.81, 93.677, 84.740, 81.532, 82.893,
     I 84.564, 87.584, 91.780, 91.272, 87.014, 87.386, 90.149, 84.917,
     J 71.266, 57.873, 51.863, 53.876, 57.909, 58.508, 57.020, 57.432,
     K 60.671, 64.667, 67.362, 67.511, 64.233, 59.035, 55.697, 56.636,
     L 59.400, 59.070, 56.522, 55.834, 55.860, 54.039, 51.976, 52.344,
     M 54.667, 56.450, 56.751, 56.769, 58.002, 60.029, 59.602, 53.134,
     N 42.926, 35.588, 33.447, 35.171, 39.379, 44.371, 47.745, 46.933,
     O 42.441, 37.879, 35.595, 36.458, 41.048, 47.300, 51.098, 50.024,
     P 45.331, 41.282, 40.082, 40.000, 39.104, 37.329, 36.632, 37.792,
     Q 39.189, 41.058, 45.214, 50.737, 54.281, 55.015, 56.138, 60.931,
     R 67.383, 69.534, 65.159, 56.372, 47.326, 44.322, 49.944, 59.696 /
C        SOLAR SPECTRUM FROM  39920 TO  41350 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB09  /
     A 67.929, 71.334, 69.905, 65.620, 59.303, 54.016, 55.880, 65.155,
     B 74.065, 76.217, 73.506, 71.406, 70.849, 69.749, 69.268, 71.380,
     C 72.721, 68.929, 61.665, 54.896, 47.420, 38.325, 32.219, 31.243,
     D 33.310, 35.358, 35.623, 36.840, 41.551, 47.499, 51.176, 50.344,
     E 45.362, 38.341, 33.130, 33.801, 40.140, 49.121, 55.385, 55.174,
     F 50.450, 46.511, 47.495, 51.883, 56.354, 59.603, 61.584, 63.215,
     G 64.603, 64.101, 59.027, 50.956, 47.633, 52.543, 58.883, 59.829,
     H 57.617, 56.727, 57.371, 57.898, 57.177, 55.129, 52.952, 52.018,
     I 52.186, 52.044, 50.269, 46.592, 42.515, 40.755, 41.887, 44.119,
     J 46.536, 48.858, 50.490, 51.919, 54.085, 54.707, 51.927, 49.449,
     K 49.865, 50.933, 50.496, 48.616, 46.717, 46.070, 46.263, 46.733,
     L 48.009, 50.187, 52.420, 53.536, 52.507, 51.380, 53.214, 56.985,
     M 60.614, 63.139, 63.999, 63.869, 65.100, 69.385, 74.743, 78.184,
     N 78.103, 74.113, 67.371, 60.849, 58.924, 62.682, 68.032, 69.117,
     O 64.604, 59.110, 55.998, 56.838, 61.778, 65.874, 65.079, 63.038,
     P 64.809, 69.911, 74.841, 76.439, 73.587, 68.853, 67.497, 72.675,
     Q 80.602, 83.422, 78.957, 72.228, 66.737, 62.842, 61.535, 63.574,
     R 69.248, 76.577, 79.922, 77.755, 73.938, 70.518, 68.003, 66.339 /
C        SOLAR SPECTRUM FROM  41360 TO  42790 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB10  /
     A 63.979, 61.098, 59.421, 58.103, 55.741, 52.549, 48.079, 42.578,
     B 38.373, 37.297, 37.455, 34.861, 30.483, 29.634, 34.734, 42.460,
```

```
     C 47.066, 45.848, 40.157, 34.290, 31.584, 30.650, 29.054, 27.788,
     D 30.427, 37.570, 44.196, 46.880, 47.848, 49.166, 49.180, 45.002,
     E 38.135, 35.055, 38.095, 41.750, 40.899, 35.722, 28.884, 24.835,
     F 28.670, 39.646, 50.310, 55.725, 57.401, 58.110, 59.406, 59.360,
     G 53.420, 43.004, 34.787, 33.697, 39.682, 47.554, 52.605, 53.632,
     H 51.001, 45.266, 37.844, 31.030, 25.936, 22.799, 21.882, 23.484,
     I 27.857, 33.447, 37.319, 39.195, 42.826, 50.398, 58.752, 63.301,
     J 61.094, 53.532, 46.046, 41.118, 37.646, 36.304, 40.426, 50.893,
     K 61.553, 65.395, 62.680, 58.087, 54.622, 51.330, 46.874, 42.870,
     L 40.547, 39.760, 40.217, 40.359, 39.559, 40.667, 46.260, 53.413,
     M 56.041, 52.566, 46.674, 41.073, 35.511, 31.231, 31.082, 35.955,
     N 45.199, 55.464, 61.802, 63.505, 61.850, 56.412, 49.388, 46.369,
     O 50.058, 56.694, 60.884, 61.030, 58.107, 54.303, 51.940, 50.508,
     P 46.749, 39.155, 31.535, 28.959, 30.973, 32.670, 31.567, 29.340,
     Q 27.275, 25.184, 24.264, 27.068, 34.296, 42.475, 47.230, 47.425,
     R 44.435, 40.538, 36.868, 33.020, 29.405, 28.753, 34.079, 44.246 /
C         SOLAR SPECTRUM FROM  42800 TO  44230 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB11  /
     A 53.780, 57.974, 56.376, 51.200, 45.308, 40.273, 35.900, 33.344,
     B 34.011, 36.858, 41.283, 47.374, 53.088, 56.201, 55.633, 50.843,
     C 43.997, 38.767, 36.248, 36.380, 40.762, 50.700, 63.371, 73.432,
     D 76.418, 70.373, 58.741, 47.034, 38.598, 34.664, 35.794, 42.084,
     E 49.973, 54.338, 53.956, 52.287, 52.778, 55.571, 59.034, 60.268,
     F 56.247, 47.362, 38.056, 32.889, 31.739, 31.734, 32.476, 35.060,
     G 39.091, 43.398, 48.131, 53.574, 58.749, 63.599, 68.971, 73.421,
     H 73.861, 69.003, 60.557, 51.865, 44.879, 42.060, 44.802, 47.950,
     I 46.882, 42.973, 39.293, 37.711, 37.137, 35.222, 32.243, 30.488,
     J 32.605, 40.429, 51.099, 57.710, 57.150, 52.992, 50.275, 49.986,
     K 49.778, 48.371, 46.421, 44.604, 42.730, 41.244, 41.565, 43.805,
     L 47.013, 48.992, 46.428, 40.595, 37.840, 42.353, 52.248, 60.529,
     M 61.566, 56.800, 52.041, 52.260, 57.077, 61.019, 60.712, 57.048,
     N 51.481, 46.352, 44.366, 44.947, 45.478, 44.944, 43.825, 42.105,
     O 39.466, 36.826, 35.907, 36.357, 35.661, 33.947, 33.690, 34.429,
     P 34.000, 32.645, 31.410, 30.281, 29.409, 29.127, 29.326, 29.869,
     Q 30.601, 31.311, 32.099, 32.779, 32.757, 32.098, 31.975, 33.484,
     R 36.048, 39.169, 43.365, 47.244, 48.214, 45.786, 41.586, 38.775 /
C         SOLAR SPECTRUM FROM  44240 TO  45670 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB12  /
     A 40.753, 46.752, 51.684, 52.597, 51.449, 50.684, 49.450, 46.747,
     B 45.369, 47.685, 50.240, 48.961, 46.693, 48.600, 53.694, 56.465,
     C 54.341, 50.722, 49.877, 51.246, 52.088, 52.765, 56.254, 63.326,
     D 69.744, 71.066, 68.349, 65.123, 62.551, 59.195, 53.705, 48.161,
     E 46.236, 47.710, 49.660, 50.799, 51.836, 54.537, 59.647, 64.707,
     F 65.844, 61.634, 55.570, 54.083, 58.781, 64.888, 69.777, 74.008,
     G 76.492, 76.226, 74.746, 74.941, 77.801, 79.619, 76.190, 67.190,
     H 55.231, 45.813, 43.141, 45.647, 49.466, 52.231, 52.221, 48.886,
     I 44.716, 42.613, 43.385, 45.968, 48.121, 48.998, 49.885, 50.707,
     J 49.893, 48.319, 48.198, 50.280, 53.830, 55.914, 54.822, 52.939,
     K 51.944, 49.438, 42.956, 34.614, 28.100, 24.503, 24.203, 27.839,
     L 34.604, 41.615, 45.324, 45.444, 45.527, 47.179, 45.756, 36.862,
     M 26.037, 20.569, 20.329, 24.263, 30.863, 35.939, 36.711, 35.693,
     N 37.256, 40.862, 44.416, 48.800, 54.182, 57.655, 58.427, 59.965,
     O 63.940, 66.820, 65.465, 59.482, 49.396, 39.422, 34.182, 35.388,
     P 42.875, 52.034, 57.595, 59.093, 57.272, 52.172, 45.493, 39.419,
     Q 35.581, 35.902, 40.354, 46.732, 53.309, 58.781, 61.785, 59.255,
     R 50.030, 41.567, 40.523, 43.584, 44.875, 42.754, 40.077, 39.941 /
C         SOLAR SPECTRUM FROM  45680 TO  47110 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB13  /
     A 40.977, 39.567, 34.955, 30.424, 31.039, 38.687, 47.480, 49.830,
     B 46.790, 44.829, 46.546, 50.415, 54.602, 57.656, 58.463, 57.276,
     C 55.621, 54.514, 53.338, 50.026, 42.817, 33.636, 27.134, 25.516,
     D 27.897, 31.392, 32.125, 29.463, 26.581, 25.956, 27.737, 31.175,
     E 34.959, 37.671, 38.641, 37.958, 36.733, 35.681, 33.877, 30.849,
     F 28.059, 27.615, 29.319, 29.375, 25.390, 20.659, 19.484, 22.297,
     G 27.282, 32.467, 35.906, 37.137, 37.895, 39.130, 39.777, 39.872,
     H 40.778, 42.317, 42.934, 40.430, 34.227, 27.701, 23.880, 22.174,
     I 21.639, 22.589, 25.184, 29.017, 32.981, 36.110, 38.580, 41.239,
     J 44.426, 46.939, 47.010, 44.165, 39.659, 35.556, 32.838, 31.546,
     K 32.676, 36.963, 42.333, 44.931, 43.704, 40.943, 37.973, 35.199,
     L 33.574, 33.339, 34.185, 36.347, 39.963, 43.964, 47.162, 48.987,
     M 48.976, 47.948, 48.004, 49.892, 51.065, 47.834, 40.489, 32.665,
     N 26.795, 24.461, 26.655, 31.928, 37.634, 41.345, 40.956, 36.827,
     O 32.110, 28.612, 26.482, 26.602, 28.831, 30.877, 30.976, 30.063,
     P 29.887, 30.305, 29.974, 28.265, 26.517, 27.066, 30.403, 34.539,
     Q 37.104, 37.598, 37.252, 37.060, 36.498, 34.167, 29.814, 24.192,
     R 18.515, 15.086, 15.040, 17.158, 20.807, 25.682, 30.352, 34.203 /
C         SOLAR SPECTRUM FROM  47120 TO  48550 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB14  /
     A 37.902, 42.531, 47.832, 50.509, 48.019, 42.616, 38.321, 37.370,
     B 40.172, 44.395, 46.132, 43.911, 38.396, 31.379, 26.275, 25.075,
     C 26.652, 28.963, 31.168, 34.168, 38.050, 40.231, 38.347, 32.741,
     D 26.199, 21.863, 20.249, 20.185, 21.726, 25.562, 30.318, 33.431,
     E 34.453, 34.959, 36.374, 37.870, 36.655, 31.966, 25.920, 21.264,
     F 20.663, 24.658, 30.263, 34.021, 34.336, 31.356, 26.926, 23.109,
     G 20.867, 20.684, 22.416, 24.878, 26.779, 27.334, 26.537, 25.210,
     H 24.013, 22.944, 21.800, 20.449, 19.290, 19.528, 21.742, 24.125,
     I 23.994, 21.559, 19.555, 18.915, 18.342, 17.335, 16.549, 16.479,
     J 17.211, 18.445, 19.294, 18.980, 17.912, 17.156, 17.103, 17.256,
     K 16.925, 15.842, 14.485, 13.683, 13.647, 13.914, 14.009, 13.770,
     L 13.456, 13.399, 13.547, 13.760, 14.060, 14.427, 14.644, 14.438,
     M 13.986, 13.749, 13.927, 14.390, 14.759, 14.822, 14.679, 14.448,
     N 14.186, 13.937, 13.754, 13.657, 13.540, 13.308, 13.053, 12.841,
     O 12.704, 12.742, 12.811, 12.662, 12.355, 12.100, 12.003, 12.014,
     P 12.067, 12.223, 12.444, 12.472, 12.164, 11.732, 11.515, 11.619,
     Q 11.873, 12.028, 11.947, 11.722, 11.399, 10.930, 10.473, 10.205,
     R 10.224, 10.694, 11.468, 12.007, 12.083, 11.905, 11.498, 10.891 /
C         SOLAR SPECTRUM FROM  48560 TO  49990 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB15  /
     A 10.575, 10.846, 11.353, 11.612, 11.411, 10.876, 10.383, 10.305,
     B 10.695, 11.245, 11.636, 11.828, 11.918, 11.865, 11.674, 11.510,
     C 11.407, 11.303, 11.216, 11.143, 11.039, 10.983, 11.004, 10.900,
     D 10.653, 10.562, 10.781, 11.186, 11.605, 11.806, 11.582, 11.056,
     E 10.567, 10.335, 10.408, 10.729, 11.165, 11.540, 11.646, 11.372,
     F 10.933, 10.524, 9.9973, 9.3783, 8.9883, 9.0163, 9.4125, 9.9179,
     G 10.278, 10.472, 10.553, 10.575, 10.519, 10.216, 9.6821, 9.1499,
     H 8.7057, 8.3894, 8.3442, 8.6241, 9.1371, 9.7184, 10.191, 10.443,
     I 10.458, 10.289, 9.9772, 9.5829, 9.3097, 9.3195, 9.4694, 9.5182,
     J 9.4326, 9.2478, 8.8197, 7.9809, 6.9996, 6.4856, 6.7462, 7.5406,
     K 8.2813, 8.7258, 9.0682, 9.1665, 8.8637, 8.4638, 8.2393, 8.1656,
     L 8.1880, 8.3578, 8.6488, 8.8980, 9.0117, 9.0659, 9.1955, 9.4207,
     M 9.5526, 9.4237, 9.1290, 8.8441, 8.6138, 8.4237, 8.2979, 8.2598,
     N 8.2859, 8.3475, 8.4533, 8.6285, 8.8310, 8.8866, 8.6750, 8.3312,
     O 8.0091, 7.7296, 7.6239, 7.8692, 8.2725, 8.4086, 8.2515, 8.0914,
     P 8.0003, 7.9367, 7.9266, 7.9580, 8.0492, 8.2376, 8.4263, 8.4811,
     Q 8.3309, 8.0263, 7.7632, 7.6987, 7.8124, 7.9390, 8.0183, 8.0816,
     R 8.0428, 7.8923, 7.6963, 7.4969, 7.4013, 7.4289, 7.4489, 7.4059 /
C         SOLAR SPECTRUM FROM  50000 TO  51430 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB16  /
     A 7.4198, 7.5261, 7.5252, 7.3239, 7.1263, 7.1423, 7.3340, 7.5049,
     B 7.5484, 7.5319, 7.5163, 7.4995, 7.5728, 7.8104, 8.0588, 8.0948,
     C 7.9140, 7.6978, 7.5116, 7.2138, 6.8063, 6.5430, 6.5232, 6.5869,
     D 6.5610, 6.3984, 6.1889, 6.0587, 6.0676, 6.1988, 6.3140, 6.2527,
     E 6.0929, 6.0277, 6.0941, 6.3031, 6.6594, 6.9398, 6.9566, 6.8310,
     F 6.7374, 6.6812, 6.6558, 6.8336, 7.2020, 7.4012, 7.2950, 7.0488,
     G 6.7966, 6.6293, 6.5868, 6.5980, 6.6007, 6.6501, 6.7627, 6.7853,
     H 6.6321, 6.4856, 6.5198, 6.6486, 6.7271, 6.7227, 6.6696, 6.6189,
     I 6.5979, 6.6188, 6.7110, 6.8343, 6.8750, 6.8250, 6.7885, 6.8266,
     J 6.8556, 6.8068, 6.8377, 7.0467, 7.2779, 7.4139, 7.4712, 7.4621,
     K 7.4071, 7.3592, 7.3372, 7.3220, 7.2938, 7.2531, 7.2052, 7.1335,
     L 7.0298, 6.8533, 6.5535, 6.2227, 6.0139, 5.9384, 5.9038, 5.8568,
```

```
     M 5.7909, 5.7326, 5.7745, 5.9608, 6.1865, 6.3681, 6.4997, 6.5437,
     N 6.4637, 6.2708, 6.0451, 5.9557, 6.0855, 6.2542, 6.2454, 6.0795,
     O 5.9102, 5.8447, 5.9218, 6.1063, 6.2895, 6.3271, 6.1097, 5.7421,
     P 5.4452, 5.2981, 5.3256, 5.4935, 5.6819, 5.8245, 5.8933, 5.9630,
     Q 6.1703, 6.4525, 6.6325, 6.6965, 6.7185, 6.6238, 6.3107, 5.9241,
     R 5.6987, 5.6651, 5.7428, 5.8790, 5.9715, 5.9618, 5.9674, 6.0754 /
C        SOLAR SPECTRUM FROM  51440 TO  52870 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB17  /
     A 6.2541, 6.4300, 6.4968, 6.4564, 6.4082, 6.3024, 6.0135, 5.6431,
     B 5.3963, 5.2989, 5.2635, 5.2227, 5.1279, 4.9315, 4.6348, 4.3168,
     C 4.0151, 3.6625, 3.2906, 3.1028, 3.1349, 3.1994, 3.2596, 3.4144,
     D 3.5949, 3.6534, 3.6296, 3.6281, 3.5876, 3.4292, 3.2659, 3.2284,
     E 3.2576, 3.3002, 3.4535, 3.7372, 4.0573, 4.3558, 4.5999, 4.7781,
     F 4.8855, 4.8999, 4.8392, 4.7624, 4.7059, 4.6981, 4.7666, 4.8453,
     G 4.8236, 4.7293, 4.6861, 4.7132, 4.7725, 4.8713, 4.9596, 4.9527,
     H 4.8957, 4.9252, 5.0736, 5.2229, 5.2505, 5.1537, 5.0156, 4.8880,
     I 4.7686, 4.6549, 4.5534, 4.4828, 4.4661, 4.5040, 4.5905, 4.7033,
     J 4.7858, 4.8334, 4.9283, 5.0377, 5.0065, 4.8471, 4.6828, 4.5586,
     K 4.4812, 4.4314, 4.3903, 4.3830, 4.4066, 4.3900, 4.2973, 4.1978,
     L 4.1462, 4.1084, 4.1495, 4.3897, 4.6859, 4.8206, 4.7938, 4.6781,
     M 4.5222, 4.3959, 4.3358, 4.2947, 4.2259, 4.1452, 4.1060, 4.1462,
     N 4.2149, 4.2549, 4.3061, 4.3742, 4.3738, 4.2718, 4.1389, 4.0405,
     O 3.9457, 3.8127, 3.7099, 3.7344, 3.8589, 3.9598, 3.9525, 3.8377,
     P 3.6708, 3.5357, 3.4929, 3.5375, 3.6381, 3.7890, 3.9671, 4.0995,
     Q 4.1421, 4.1302, 4.1235, 4.1623, 4.2506, 4.2948, 4.2231, 4.0993,
     R 3.9680, 3.9475, 4.1958, 4.5131, 4.6101, 4.5130, 4.3474, 4.1749 /
C        SOLAR SPECTRUM FROM  52880 TO  54310 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB18  /
     A 4.0467, 3.9956, 4.0078, 4.0374, 4.0255, 3.9379, 3.8192, 3.7529,
     B 3.7675, 3.8260, 3.8654, 3.8518, 3.8148, 3.8028, 3.8098, 3.7934,
     C 3.7660, 3.7944, 3.8689, 3.8978, 3.8856, 3.8923, 3.8570, 3.6940,
     D 3.4693, 3.3222, 3.2824, 3.2887, 3.3039, 3.3222, 3.3313, 3.3326,
     E 3.3482, 3.3807, 3.4188, 3.4602, 3.4972, 3.5151, 3.5155, 3.5165,
     F 3.5258, 3.5406, 3.5478, 3.5345, 3.5339, 3.5820, 3.6396, 3.6448,
     G 3.5872, 3.5112, 3.4804, 3.5257, 3.6238, 3.7290, 3.8023, 3.8024,
     H 3.7268, 3.6578, 3.6439, 3.6422, 3.6373, 3.6397, 3.6410, 3.6494,
     I 3.6608, 3.6251, 3.5212, 3.4020, 3.2845, 3.1230, 2.9483, 2.8515,
     J 2.8432, 2.8638, 2.8967, 2.9505, 3.0025, 3.0552, 3.1106, 3.1178,
     K 3.0596, 2.9854, 2.9316, 2.8903, 2.8590, 2.8500, 2.8450, 2.8121,
     L 2.7626, 2.7424, 2.7667, 2.8024, 2.8165, 2.8111, 2.8128, 2.8569,
     M 2.9659, 3.1062, 3.1990, 3.2128, 3.2088, 3.2391, 3.2661, 3.2364,
     N 3.1173, 2.9094, 2.6952, 2.5324, 2.3959, 2.2953, 2.2510, 2.2245,
     O 2.1811, 2.1301, 2.1482, 2.3257, 2.5856, 2.7226, 2.6495, 2.4508,
     P 2.2444, 2.0850, 1.9891, 1.9843, 2.0816, 2.2233, 2.3248, 2.3551,
     Q 2.3479, 2.3606, 2.4296, 2.5361, 2.6128, 2.6216, 2.6069, 2.6196,
     R 2.6464, 2.6427, 2.5823, 2.4682, 2.3320, 2.2405, 2.2637, 2.3973 /
C        SOLAR SPECTRUM FROM  54320 TO  55750 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB19  /
     A 2.5524, 2.6891, 2.8508, 3.0103, 3.0681, 3.0064, 2.9114, 2.8609,
     B 2.8517, 2.8374, 2.7894, 2.7288, 2.7138, 2.7729, 2.8707, 2.9536,
     C 2.9953, 2.9911, 2.9398, 2.8550, 2.7732, 2.7303, 2.7366, 2.7650,
     D 2.7705, 2.7374, 2.6830, 2.6218, 2.5663, 2.5341, 2.5351, 2.5681,
     E 2.6124, 2.6305, 2.6024, 2.5431, 2.4840, 2.4546, 2.4684, 2.5100,
     F 2.5445, 2.5532, 2.5564, 2.5889, 2.6616, 2.7553, 2.8466, 2.9290,
     G 2.9958, 3.0175, 2.9774, 2.8990, 2.8001, 2.6927, 2.6171, 2.5931,
     H 2.5809, 2.5276, 2.4284, 2.3365, 2.3162, 2.3855, 2.4872, 2.5455,
     I 2.5773, 2.6809, 2.9720, 3.5757, 4.4006, 5.0044, 5.0295, 4.5135,
     J 3.7071, 2.9059, 2.3600, 2.1418, 2.1119, 2.0871, 2.0301, 2.0043,
     K 2.0361, 2.0963, 2.1520, 2.1878, 2.1955, 2.1864, 2.1899, 2.2170,
     L 2.2574, 2.2895, 2.2783, 2.2148, 2.1641, 2.2343, 2.4726, 2.8119,
     M 3.1288, 3.2984, 3.2206, 2.8859, 2.4473, 2.1436, 2.0729, 2.1391,
     N 2.2171, 2.2580, 2.2654, 2.2481, 2.2103, 2.1657, 2.1356, 2.1321,
     O 2.1438, 2.1461, 2.1396, 2.1460, 2.1588, 2.1581, 2.1481, 2.1343,
     P 2.1101, 2.0754, 2.0400, 2.0121, 1.9930, 1.9799, 1.9699, 1.9613,
     Q 1.9537, 1.9454, 1.9312, 1.9058, 1.8726, 1.8470, 1.8465, 1.8693,
     R 1.8844, 1.8635, 1.8143, 1.7618, 1.7188, 1.6853, 1.6656, 1.6708 /
C        SOLAR SPECTRUM FROM  55760 TO  57190 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB20  /
     A 1.7036, 1.7519, 1.8120, 1.9015, 2.0124, 2.0980, 2.1385, 2.1481,
     B 2.1347, 2.1086, 2.0953, 2.1062, 2.1095, 2.0685, 2.0001, 1.9461,
     C 1.9194, 1.9088, 1.9023, 1.8977, 1.9049, 1.9300, 1.9588, 1.9635,
     D 1.9357, 1.9019, 1.8887, 1.8939, 1.9018, 1.9038, 1.8975, 1.8747,
     E 1.8289, 1.7716, 1.7303, 1.7330, 1.7900, 1.8782, 1.9548, 1.9907,
     F 1.9807, 1.9430, 1.9173, 1.9218, 1.9203, 1.8717, 1.7832, 1.6965,
     G 1.6389, 1.6077, 1.5924, 1.5818, 1.5583, 1.5142, 1.4616, 1.4237,
     H 1.4252, 1.4834, 1.5970, 1.7410, 1.8771, 1.9784, 2.0451, 2.0872,
     I 2.0909, 2.0384, 1.9573, 1.9002, 1.8824, 1.8663, 1.8193, 1.7540,
     J 1.6874, 1.6222, 1.5726, 1.5450, 1.5290, 1.5312, 1.5699, 1.6411,
     K 1.7186, 1.7678, 1.7546, 1.6623, 1.5115, 1.3588, 1.2605, 1.2348,
     L 1.2611, 1.3091, 1.3588, 1.3884, 1.3800, 1.3482, 1.3224, 1.3159,
     M 1.3437, 1.4142, 1.4950, 1.5443, 1.5521, 1.5282, 1.4902, 1.4606,
     N 1.4465, 1.4398, 1.4399, 1.4544, 1.4760, 1.4781, 1.4506, 1.4229,
     O 1.4185, 1.4221, 1.4119, 1.3908, 1.3779, 1.3813, 1.3933, 1.4087,
     P 1.4268, 1.4417, 1.4408, 1.4188, 1.3861, 1.3548, 1.3261, 1.2980,
     Q 1.2769, 1.2731, 1.2856, 1.3002, 1.3056, 1.2987, 1.2817, 1.2590,
     R 1.2291, 1.1868, 1.1428, 1.1183, 1.1141, 1.1120, 1.1009, 1.0797 /
C    SOLAR SPECTRUM FROM  57200 TO  57490 CM-1, IN STEPS OF 10 CM-1
      DATA SUNB21  /
     A 1.0523, 1.0284, 1.0251, 1.0577, 1.1195, 1.1791, 1.2061, 1.2013,
     B 1.1936, 1.2000, 1.2040, 1.1824, 1.1489, 1.1400, 1.1539, 1.1629,
     C 1.1617, 1.1586, 1.1564, 1.1572, 1.1565, 1.1399, 1.1037, 1.0627,
     D 1.0341, 1.0223, 1.0199, 1.0188, 1.0174, 1.0163  /
      END
```

```
      REAL FUNCTION SOLR_IRRAD(WAVLEN,IDAY)

C      PURPOSE          CALCULATE SOLAR SPECTRAL IRRADIANCE INCIDENT AT
C                       TOP OF EARTH'S ATMOSPHERE AT A GIVEN WAVELENGTH.
C                       ALLOWANCE IS MADE FOR THE ELLIPTICAL ORBIT
C                       OF THE EARTH ABOUT THE SUN.
C                       BASED ON LOWTRAN7 SUBROUTINE "SOURCE"
C      INPUTS
C      WAVLEN           DESIRED WAVELENGTH IN MICRONS
C      DAY              DAY OF YEAR
C      OUTPUT
C      SOLAR            SOLAR SPECTRAL IRRADIANCE (MILLIWATTS/CM**2/MICR
C-----------------------------------------------------------------

      DIMENSION NDAY(13),RAT(13)
      DATA NDAY/1,32,60,91,121,152,182,213,244,274,305,335,366/
      DATA RAT/1.034,1.030,1.019,1.001,.985,.972,.967,.971,.982,
     1    .998,1.015,1.029,1.034/
CHANGE*** 2 AUG 89 LG
c     WRITE(*,*)'ENTERING SOLAR ',wavlen, iday
      VV=10000.0/WAVLEN
      V=10000./VV
      IF(VV.LE.0.) V = 1.0E+38
C
C     SUN ELLIPTIC ORBIT FACTOR
C
      FORBIT=0.0
      IF(IDAY.GT.0 .AND. IDAY.LT.367) GO TO 55
      FORBIT = 1.0
      GO TO 90
55    CONTINUE
      DO 60 I=1,13
      IF(NDAY(I).EQ.IDAY) GO TO 80
      IF(NDAY(I).GT.IDAY) GO TO 70
60    CONTINUE
70    FORBIT=RAT(I-1)+(IDAY-NDAY(I-1))*(RAT(I)-RAT(I-1))/(NDAY(I)
     1   -NDAY(I-1))
      GO TO 90
80    FORBIT=RAT(I)
90    CONTINUE
C
C     SOLAR INTENSITY
C
      SS = SUN(VV)*FORBIT
C     CONVERT W/M-2-MICRON TO MW/CM-2-MICRON
C     SOLR_IRRAD = SS*.1
      SOLR_IRRAD = SS
C
      RETURN
      END
C
      FUNCTION   SUN(V)
C
C        EVALUATES THE EXTRA-TERRESTRIAL SOLAR IRRADIANCE
C
C        INPUT:  V  =  FREQUENCY    (CM-1)
C                VALID RANGE   0 TO 57490 (CM-1)
C                    (EQUIVALENT TO WAVELENGTHS > 0.174 MICROMETERS)
C
C        OUTPUT:  SUN  =  SOLAR IRRADIANCE  (WATTS M-2 MICROMETER-1)
C
C        WRITES A WARNING MESSAGE TO TAPE6  &  RETURNS  SUN = 0
C            IF THE INPUT FREQUENCY IS OUT OF RANGE
C
C       USES  BLOCK DATA SOLAR   WHICH CONTAINS THE VALUES FOR SOLARA +
C
      COMMON /SUNDAT/ SOLARA(1440), SOLARB(2910)
      DATA  A, B / 3.50187E-13, 3.93281 /
C
C       WM, W0, W1, W2  ARE STATEMENT FUNKTIONS USED BY
C           THE 4 POINT LAGRANGE INTERPOLATION
      WM(P) = P*(P - 1)*(P - 2)
      W0(P) = 3*(P**2 - 1)*(P - 2)
      W1(P) = 3*P*(P + 1)*(P - 2)
      W2(P) = P*(P**2 - 1)
C
C           IF  V  IS TOO SMALL,  WRITE WARNING  +  RETURN SUN = 0
      IF(V .LT. 0.0) THEN
        SUN = 0.0
C       WRITE(6, 900)'MAS WAVENUMBER LESS THAN ZERO ',V
        RETURN
C
      ELSEIF( V .GE. 0.0   .AND.  V .LT. 100.0 ) THEN
C         FOR LOW FREQUENCIES USE A POWER LAW APPROXIMATION
        SUN = A*V**B
        RETURN
C
      ELSEIF( V .GE. 100.0   .AND.  V .LT. 28420.0 ) THEN
C         USE  4 POINT INTERPOLATION  ON  ARRAY  SOLARA
C               WHICH IS AT  20 CM-1  SPACING  FROM 0 TO 28720 CM-1
        I = 1 + INT(V/20.0)
        P = MOD(V, 20.0)/20.0
        SUN = ( W2(P)*SOLARA(I+2) - W1(P)*SOLARA(I+1) +
     +             W0(P)*SOLARA(I) - WM(P)*SOLARA(I-1) ) / 6
        RETURN
C
      ELSEIF( V .GE. 28420.0   .AND.  V .LE. 57470.0 ) THEN
C         USE  4 POINT INTERPOLATION  ON  ARRAY  SOLARB
C             WHICH IS AT  10 CM-1  SPACING  FROM 28400 TO 57490 CM-1
        I = INT(V/10.0) - 2839
        P = MOD(V, 10.0)/10.0
        SUN = ( W2(P)*SOLARB(I+2) - W1(P)*SOLARB(I+1) +
     +             W0(P)*SOLARB(I) - WM(P)*SOLARB(I-1) ) / 6
        RETURN
C
      ELSEIF( V .GT. 57470.0 ) THEN
C           IF  V  IS TOO LARGE,  WRITE WARNING  +  RETURN SUN = 0
        SUN = 0.0
        WRITE(6, 900)'MAS WAVENUMBER TOO LARGE ',V
        RETURN
C
      ENDIF
C
      RETURN
  900 FORMAT('0 *****  WARNING - INPUT FREQUENCY = ', 1PG12.5, 'CM-1',
     +  /, '   OUTSIDE VALID RANGE OF 0 TO 57470 CM-1     *******', / )
      END
      BLOCK DATA SOLARS
C>    BLOCK DATA
C
C
C     COMMON /SUNDAT/   SOLARA(1440), SOLARB(2910)
      COMMON /SUNDAT/   SUNA01( 41),SUNA02(144),SUNA03(144),SUNA04(144),
     A      SUNA05(144),SUNA06(144),SUNA07(144),SUNA08(144),SUNA09(144),
     B      SUNA10(144),SUNA11(103),SUNB01(144),SUNB02(144),SUNB03(144),
     C      SUNB04(144),SUNB05(144),SUNB06(144),SUNB07(144),SUNB08(144),
     D      SUNB09(144),SUNB10(144),SUNB11(144),SUNB12(144),SUNB13(144),
     E      SUNB14(144),SUNB15(144),SUNB16(144),SUNB17(144),SUNB18(144),
     F      SUNB19(144),SUNB20(144),SUNB21( 30)
C
```

```
C        SOLAR SPECTRUM FROM      0 TO    800 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA01  /
    A 0.0000E+00, 4.5756E-08, 7.0100E-07, 3.4580E-06, 1.0728E-05,
    B 2.5700E-05, 5.2496E-05, 9.6003E-05, 1.6193E-04, 2.5766E-04,
    C 3.9100E-04, 5.6923E-04, 8.0203E-04, 1.1006E-03, 1.4768E-03,
    D 1.9460E-03, 2.5213E-03, 3.2155E-03, 4.0438E-03, 5.0229E-03,
    E 6.1700E-03, 7.5145E-03, 9.0684E-03, 1.0853E-02, 1.2889E-02,
    F 1.5213E-02, 1.7762E-02, 2.0636E-02, 2.3888E-02, 2.7524E-02,
    G 3.1539E-02, 3.5963E-02, 4.0852E-02, 4.6236E-02, 5.2126E-02,
    H 5.8537E-02, 6.5490E-02, 7.3017E-02, 8.1169E-02, 9.0001E-02,
    I 9.9540E-02 /
C        SOLAR SPECTRUM FROM    820 TO   3680 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA02  /
    A .10980, .12080, .13260, .14520, .15860, .17310, .18850, .20490,
    B .22240, .24110, .26090, .28200, .30430, .32790, .35270, .37890,
    C .40650, .43550, .46600, .49800, .53160, .56690, .60390, .64260,
    D .68320, .72560, .76990, .81620, .86440, .91470, .96710, 1.0220,
    E 1.0780, 1.1370, 1.1990, 1.2630, 1.3290, 1.3990, 1.4710, 1.5460,
    F 1.6250, 1.7060, 1.7910, 1.8800, 1.9710, 2.0670, 2.1660, 2.2680,
    G 2.3740, 2.4840, 2.5970, 2.7140, 2.8350, 2.9600, 3.0890, 3.2210,
    H 3.3570, 3.4980, 3.6420, 3.7900, 3.9440, 4.1040, 4.2730, 4.4450,
    I 4.6150, 4.7910, 4.9830, 5.1950, 5.4210, 5.6560, 5.8930, 6.1270,
    J 6.3560, 6.5820, 6.8080, 7.0360, 7.2700, 7.5170, 7.7890, 8.0910,
    K 8.4070, 8.7120, 8.9900, 9.2490, 9.5000, 9.7550, 10.010, 10.250,
    L 10.480, 10.700, 10.950, 11.230, 11.550, 11.900, 12.250, 12.600,
    M 12.930, 13.250, 13.530, 13.780, 14.040, 14.320, 14.660, 15.070,
    N 15.530, 16.011, 16.433, 16.771, 17.077, 17.473, 17.964, 18.428,
    O 18.726, 18.906, 19.141, 19.485, 19.837, 20.160, 20.509, 21.024,
    P 21.766, 22.568, 23.190, 23.577, 23.904, 24.335, 24.826, 25.236,
    Q 25.650, 26.312, 27.208, 27.980, 28.418, 28.818, 29.565, 30.533,
    R 31.247, 31.667, 32.221, 33.089, 33.975, 34.597, 35.004, 35.395 /
C        SOLAR SPECTRUM FROM   3700 TO   6560 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA03  /
    A 36.026, 36.985, 37.890, 38.401, 38.894, 39.857, 40.926, 41.570,
    B 42.135, 43.083, 44.352, 45.520, 45.982, 46.281, 48.335, 51.987,
    C 54.367, 54.076, 52.174, 50.708, 52.153, 55.707, 56.549, 54.406,
    D 53.267, 56.084, 61.974, 64.406, 60.648, 55.146, 53.067, 57.476,
    E 64.645, 68.348, 69.055, 69.869, 70.943, 71.662, 72.769, 74.326,
    F 75.257, 74.883, 73.610, 73.210, 74.886, 78.042, 80.204, 80.876,
    G 82.668, 84.978, 86.244, 88.361, 91.998, 95.383, 98.121, 100.29,
    H 100.64, 99.997, 101.82, 105.06, 107.50, 109.99, 112.45, 113.90,
    I 113.79, 119.23, 121.96, 124.58, 127.14, 125.19, 124.37, 125.00,
    J 127.88, 130.67, 131.98, 133.74, 136.69, 136.18, 135.02, 137.44,
    K 138.44, 137.25, 136.35, 142.60, 144.54, 148.37, 151.90, 151.55,
    L 155.35, 157.59, 159.70, 162.28, 168.44, 171.43, 169.82, 170.33,
    M 172.28, 176.68, 181.92, 186.06, 187.85, 186.00, 189.82, 189.35,
    N 192.86, 202.00, 209.63, 205.76, 212.88, 215.63, 216.51, 219.20,
    O 220.29, 221.12, 227.12, 229.97, 233.23, 233.95, 234.52, 234.45,
    P 235.77, 239.80, 243.11, 241.19, 242.34, 243.69, 242.84, 246.19,
    Q 246.11, 246.76, 251.75, 255.38, 258.74, 260.26, 263.40, 268.68,
    R 271.81, 272.95, 273.93, 274.74, 274.43, 279.69, 287.76, 287.72 /
C        SOLAR SPECTRUM FROM   6580 TO   9440 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA04  /
    A 287.96, 290.01, 291.92, 295.28, 296.78, 300.46, 302.19, 299.14,
    B 301.43, 305.68, 309.29, 310.63, 313.24, 314.61, 309.58, 318.81,
    C 320.54, 321.62, 328.58, 331.66, 337.20, 345.62, 345.54, 342.96,
    D 344.38, 346.23, 349.17, 351.79, 354.71, 356.97, 358.29, 362.29,
    E 364.15, 364.97, 367.81, 368.98, 369.07, 372.17, 377.79, 381.25,
    F 384.22, 388.66, 393.58, 396.98, 398.72, 400.61, 404.06, 408.23,
    G 412.47, 415.58, 416.17, 416.53, 419.55, 425.88, 433.30, 437.73,
    H 438.13, 439.79, 441.51, 438.71, 434.25, 437.54, 448.95, 448.86,
    I 439.46, 437.10, 439.34, 444.33, 455.00, 467.05, 473.04, 469.64,
    J 467.53, 473.78, 477.50, 477.50, 480.96, 483.94, 482.19, 479.08,
    K 482.09, 493.43, 498.40, 492.05, 489.53, 493.34, 495.51, 496.52,
    L 499.57, 504.65, 509.68, 512.00, 512.05, 512.31, 515.00, 520.70,
    M 527.30, 531.88, 532.16, 530.48, 532.33, 539.26, 548.57, 553.00,
    N 548.96, 546.05, 551.00, 556.41, 557.21, 557.85, 560.95, 564.02,
    O 565.57, 566.38, 567.88, 571.48, 576.68, 581.54, 586.51, 593.62,
    P 600.70, 602.79, 601.39, 603.00, 606.88, 605.95, 600.97, 600.79,
    Q 607.21, 612.87, 614.13, 614.39, 616.61, 620.53, 625.19, 629.78,
    R 633.79, 637.31, 640.47, 642.53, 642.62, 641.93, 643.11, 646.68 /
C        SOLAR SPECTRUM FROM   9460 TO  12320 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA05  /
    A 650.57, 654.30, 660.95, 672.10, 682.31, 684.89, 682.20, 682.53,
    B 687.79, 691.42, 689.62, 688.14, 693.71, 703.25, 708.07, 706.22,
    C 704.64, 708.97, 717.35, 725.43, 731.08, 734.17, 735.41, 736.60,
    D 739.34, 742.90, 745.04, 744.29, 742.44, 749.53, 755.70, 758.82,
    E 766.31, 761.53, 762.09, 769.68, 764.18, 763.75, 768.88, 762.69,
    F 753.93, 762.38, 765.79, 772.19, 760.67, 762.10, 766.76, 766.98,
    G 769.35, 773.50, 766.84, 763.60, 773.82, 777.18, 779.61, 792.48,
    H 797.54, 787.81, 793.75, 805.96, 804.77, 806.62, 821.72, 830.28,
    I 827.54, 831.06, 830.20, 826.22, 823.28, 822.18, 833.92, 854.58,
    J 859.80, 862.56, 871.16, 875.16, 867.67, 863.87, 883.30, 893.40,
    K 897.74, 905.24, 905.38, 911.07, 930.21, 939.24, 934.74, 935.15,
    L 942.38, 948.13, 947.00, 951.88, 960.12, 951.88, 954.22, 959.07,
    M 963.36, 980.16, 983.66, 978.76, 979.38, 985.24, 977.08, 919.94,
    N 899.68, 962.91, 997.17, 999.93, 995.65, 999.93, 1014.9, 951.57,
    O 893.52, 955.14, 1003.1, 990.13, 978.79, 1011.2, 1034.7, 1031.9,
    P 1029.9, 1039.7, 1045.5, 1044.1, 1049.6, 1056.1, 1049.8, 1038.0,
    Q 1051.9, 1072.2, 1075.5, 1077.0, 1079.3, 1078.0, 1075.7, 1079.7,
    R 1081.0, 1069.8, 1078.4, 1104.3, 1111.4, 1111.7, 1117.6, 1119.6 /
C        SOLAR SPECTRUM FROM  12340 TO  15200 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA06  /
    A 1109.3, 1100.6, 1112.9, 1122.7, 1119.5, 1123.9, 1136.1, 1143.7,
    B 1140.5, 1141.2, 1151.5, 1148.7, 1138.3, 1141.0, 1150.6, 1160.1,
    C 1170.6, 1177.7, 1179.8, 1181.7, 1182.4, 1179.8, 1181.8, 1188.3,
    D 1190.0, 1191.4, 1197.0, 1196.0, 1192.2, 1200.6, 1210.4, 1209.1,
    E 1207.5, 1205.3, 1193.3, 1192.9, 1220.0, 1243.3, 1245.4, 1241.5,
    F 1240.2, 1241.1, 1244.0, 1248.5, 1253.2, 1257.1, 1259.9, 1261.9,
    G 1263.6, 1265.7, 1269.6, 1277.0, 1284.2, 1284.4, 1282.7, 1287.2,
    H 1286.8, 1272.3, 1262.2, 1270.7, 1288.8, 1304.8, 1311.8, 1312.2,
    I 1314.4, 1320.2, 1326.2, 1328.4, 1325.3, 1322.5, 1325.4, 1334.6,
    J 1346.4, 1354.0, 1353.7, 1347.3, 1338.3, 1331.0, 1329.7, 1338.0,
    K 1351.9, 1363.0, 1368.8, 1372.0, 1375.9, 1382.1, 1387.8, 1388.8,
    L 1388.2, 1392.2, 1401.7, 1412.9, 1418.2, 1410.7, 1395.9, 1385.7,
    M 1388.1, 1405.0, 1424.0, 1428.1, 1422.2, 1423.6, 1434.5, 1445.2,
    N 1450.7, 1451.8, 1451.5, 1453.9, 1459.9, 1466.9, 1471.3, 1469.4,
    O 1462.5, 1460.4, 1468.9, 1481.8, 1490.8, 1495.3, 1497.9, 1500.7,
    P 1505.2, 1510.0, 1512.3, 1512.7, 1515.6, 1521.6, 1524.2, 1520.7,
    Q 1520.3, 1531.6, 1545.7, 1548.2, 1541.7, 1542.2, 1553.6, 1563.6,
    R 1563.6, 1559.9, 1561.3, 1569.9, 1581.6, 1577.6, 1529.7, 1447.0 /
C        SOLAR SPECTRUM FROM  15220 TO  18080 CM-1,  IN STEPS OF 20 CM-
     DATA SUNA07  /
    A 1396.9, 1428.7, 1506.4, 1567.1, 1594.0, 1606.1, 1613.5, 1609.0,
    B 1588.6, 1567.8, 1567.3, 1587.2, 1610.2, 1624.4, 1630.2, 1630.9,
    C 1628.1, 1622.3, 1616.9, 1618.9, 1631.6, 1648.1, 1658.2, 1659.7,
    D 1658.1, 1658.0, 1659.4, 1660.4, 1659.2, 1653.7, 1645.3, 1642.1,
    E 1652.7, 1674.2, 1694.1, 1700.6, 1703.4, 1697.6, 1654.5, 1644.4,
    F 1661.6, 1676.3, 1707.7, 1703.1, 1710.8, 1732.3, 1716.5, 1719.6,
    G 1729.6, 1683.1, 1628.5, 1683.5, 1727.0, 1707.8, 1689.4, 1698.4,
    H 1733.1, 1737.8, 1714.1, 1734.6, 1750.1, 1750.1, 1760.3, 1764.3,
    I 1765.3, 1769.4, 1779.9, 1793.0, 1765.1, 1729.4, 1745.9, 1753.4,
    J 1758.1, 1775.0, 1768.4, 1767.9, 1789.5, 1806.6, 1799.3, 1782.6,
    K 1779.3, 1792.1, 1809.7, 1808.0, 1794.4, 1818.6, 1774.2, 1648.5,
    L 1674.3, 1789.3, 1847.2, 1848.3, 1812.9, 1796.4, 1840.3, 1868.3,
    M 1864.6, 1873.2, 1872.2, 1856.0, 1845.0, 1842.4, 1823.9, 1795.1,
    N 1819.6, 1861.5, 1857.7, 1838.6, 1840.5, 1863.5, 1876.8, 1884.4,
    O 1894.9, 1875.2, 1821.2, 1779.4, 1810.2, 1855.3, 1831.8, 1837.3,
    P 1882.3, 1866.4, 1819.6, 1804.8, 1831.4, 1861.6, 1867.1, 1862.9,
    Q 1851.9, 1834.7, 1835.2, 1845.1, 1831.9, 1803.6, 1792.5, 1821.8,
```

```
     R 1845.8, 1832.3, 1847.6, 1894.2, 1909.2, 1901.0, 1891.2, 1869.9 /
C         SOLAR SPECTRUM FROM  18100 TO  20960 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA08  /
     A 1854.4, 1865.8, 1873.7, 1868.8, 1881.7, 1897.1, 1884.2, 1856.2,
     B 1840.6, 1855.1, 1885.3, 1903.6, 1900.1, 1887.4, 1887.7, 1879.0,
     C 1844.5, 1844.1, 1877.1, 1847.3, 1785.1, 1792.6, 1848.7, 1894.4,
     D 1908.8, 1892.8, 1867.4, 1885.6, 1959.9, 1971.9, 1895.8, 1883.5,
     E 1917.6, 1853.8, 1793.0, 1875.6, 1974.0, 1975.7, 1943.9, 1926.4,
     F 1914.4, 1902.7, 1882.5, 1813.3, 1710.8, 1717.9, 1859.7, 1965.1,
     G 1970.1, 1941.4, 1902.5, 1852.0, 1836.3, 1879.3, 1901.6, 1862.9,
     H 1839.1, 1840.9, 1780.0, 1684.9, 1677.3, 1718.7, 1697.3, 1684.3,
     I 1784.5, 1898.0, 1910.3, 1877.2, 1866.6, 1862.6, 1860.3, 1899.7,
     J 1971.0, 1999.9, 1970.9, 1936.5, 1922.8, 1922.8, 1924.0, 1917.2,
     K 1912.0, 1926.2, 1959.7, 1995.4, 1995.9, 1938.8, 1883.5, 1894.7,
     L 1933.3, 1935.1, 1899.3, 1852.7, 1820.2, 1821.5, 1865.2, 1935.5,
     M 1966.1, 1919.6, 1881.2, 1931.5, 2015.6, 2050.0, 2021.4, 1960.8,
     N 1938.2, 1997.0, 2051.0, 2003.4, 1912.1, 1880.2, 1895.2, 1898.0,
     O 1898.8, 1938.3, 1994.2, 2010.0, 1982.4, 1948.8, 1927.3, 1911.6,
     P 1877.7, 1791.6, 1679.8, 1645.0, 1727.3, 1845.2, 1926.2, 1973.4,
     Q 2005.2, 2021.6, 2021.8, 2025.7, 2054.3, 2086.5, 2082.6, 2052.9,
     R 2047.1, 2070.2, 2072.4, 2038.1, 2020.2, 2049.9, 2074.0, 2038.1 /
C         SOLAR SPECTRUM FROM  20980 TO  23840 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA09  /
     A 1978.6, 1963.5, 1996.8, 2037.5, 2057.5, 2048.2, 2018.4, 1999.2,
     B 2011.4, 2039.5, 2056.0, 2040.2, 1981.8, 1911.4, 1891.8, 1938.3,
     C 1991.7, 2005.5, 2000.8, 2011.3, 2022.7, 1997.5, 1947.7, 1936.3,
     D 1986.6, 2037.9, 2032.8, 1995.7, 1984.0, 2012.0, 2055.5, 2091.6,
     E 2106.5, 2094.9, 2070.4, 2052.8, 2046.7, 2043.8, 2035.5, 2016.6,
     F 1988.4, 1973.3, 1999.0, 2057.4, 2103.8, 2109.4, 2089.4, 2068.5,
     G 2051.8, 2031.2, 2005.9, 1986.7, 1981.5, 1979.4, 1964.1, 1943.6,
     H 1951.8, 2007.3, 2083.2, 2139.1, 2158.0, 2143.3, 2103.2, 2050.9,
     I 2001.9, 1974.5, 1988.0, 2037.8, 2075.1, 2050.6, 1971.5, 1884.5,
     J 1828.5, 1820.9, 1866.4, 1935.3, 1974.2, 1958.7, 1925.1, 1920.2,
     K 1949.7, 1984.6, 1996.4, 1966.4, 1884.8, 1781.9, 1726.8, 1759.4,
     L 1817.4, 1800.4, 1692.6, 1593.2, 1598.6, 1700.3, 1823.8, 1909.7,
     M 1937.7, 1902.5, 1822.4, 1737.8, 1683.2, 1666.8, 1682.7, 1715.3,
     N 1734.1, 1712.4, 1668.2, 1655.0, 1698.1, 1727.2, 1636.9, 1415.7,
     O 1204.2, 1155.8, 1278.4, 1450.0, 1560.5, 1595.1, 1587.8, 1570.6,
     P 1565.8, 1590.3, 1640.5, 1688.4, 1708.1, 1703.6, 1700.7, 1718.5,
     Q 1749.0, 1772.2, 1772.5, 1745.2, 1690.2, 1624.9, 1589.0, 1618.5,
     R 1701.3, 1783.2, 1816.4, 1800.7, 1765.0, 1734.1, 1714.6, 1705.0 /
C         SOLAR SPECTRUM FROM  23860 TO  26720 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA10  /
     A 1701.6, 1696.6, 1682.0, 1661.4, 1657.2, 1693.0, 1763.2, 1826.5,
     B 1841.6, 1806.1, 1755.6, 1725.8, 1724.2, 1736.8, 1749.0, 1756.1,
     C 1759.5, 1762.1, 1770.2, 1791.7, 1826.8, 1848.9, 1819.6, 1720.7,
     D 1595.5, 1513.9, 1522.5, 1602.0, 1706.2, 1793.4, 1837.9, 1820.3,
     E 1738.3, 1631.1, 1553.1, 1539.2, 1574.3, 1623.9, 1660.6, 1676.8,
     F 1673.1, 1652.9, 1626.4, 1606.7, 1604.2, 1620.9, 1654.5, 1701.2,
     G 1752.2, 1796.2, 1822.8, 1827.4, 1808.5, 1767.0, 1713.9, 1667.3,
     H 1643.7, 1643.5, 1652.5, 1655.3, 1638.7, 1592.2, 1506.4, 1377.3,
     I 1209.5, 1010.5, 807.59, 666.84, 664.53, 835.23, 1099.6, 1330.7,
     J 1423.2, 1363.7, 1194.1, 961.77, 725.04, 551.29, 504.01, 596.30,
     K 775.15, 975.62, 1150.2, 1287.2, 1386.1, 1447.5, 1473.7, 1468.5,
     L 1435.2, 1376.9, 1296.0, 1195.5, 1085.3, 985.40, 917.25, 894.59,
     M 910.86, 951.53, 1001.7, 1046.4, 1070.7, 1061.2, 1021.2, 977.16,
     N 959.15, 982.06, 1020.5, 1032.6, 983.44, 879.83, 762.66, 675.28,
     O 643.33, 662.65, 721.49, 808.35, 913.24, 1027.0, 1139.9, 1236.2,
     P 1293.2, 1287.1, 1210.4, 1102.1, 1021.6, 1022.8, 1109.3, 1232.6,
     Q 1337.0, 1383.1, 1372.8, 1324.7, 1257.7, 1188.8, 1133.5, 1106.5,
     R 1113.7, 1136.8, 1147.9, 1121.4, 1054.1, 968.10, 889.19, 837.87 /
C         SOLAR SPECTRUM FROM  26740 TO  28780 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA11  /
     A 817.64, 823.72, 851.04, 896.53, 959.85, 1041.2, 1137.6, 1231.2,
     B 1294.4, 1299.9, 1241.2, 1155.0, 1092.0, 1097.1, 1170.2, 1263.5,
     C 1322.4, 1307.4, 1233.6, 1146.1, 1090.8, 1092.5, 1134.6, 1188.9,
     D 1228.9, 1245.5, 1248.5, 1250.3, 1260.5, 1274.6, 1279.5, 1261.8,
     E 1214.3, 1145.4, 1069.6, 1001.4, 952.52, 930.48, 941.68, 990.34,
     F 1064.4, 1135.2, 1171.5, 1149.1, 1076.3, 984.35, 906.25, 868.17,
     G 873.75, 915.33, 984.41, 1067.2, 1137.1, 1163.1, 1115.5, 990.55,
     H 830.93, 692.29, 627.44, 654.10, 739.24, 838.88, 911.69, 941.90,
     I 944.42, 939.58, 946.10, 970.23, 1005.2, 1042.4, 1073.8, 1097.0,
     J 1114.3, 1128.8, 1142.9, 1153.4, 1152.4, 1131.5, 1084.2, 1016.7,
     K 945.95, 890.37, 866.15, 876.54, 913.13, 966.10, 1025.4, 1080.2,
     L 1119.0, 1102.7, 1243.5, 1209.9, 1079.2, 852.20, 956.80, 842.31,
     M 897.44, 1081.8, 914.23, 993.09, 1049.8, 844.95, 839.16/
C         SOLAR SPECTRUM FROM  28400 TO  29830 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB01  /
     A 876.54, 892.17, 913.13, 938.18, 966.10, 995.62, 1025.4, 1054.1,
     B 1080.2, 1102.1, 1119.0, 1132.2, 1102.7, 1159.3, 1243.5, 1238.3,
     C 1209.9, 1196.2, 1079.2, 895.60, 852.20, 935.59, 956.80, 897.09,
     D 842.31, 821.15, 897.44, 1042.7, 1081.8, 988.79, 914.23, 929.38,
     E 993.09, 1041.9, 1049.8, 984.33, 844.95, 770.76, 839.16, 939.65,
     F 1026.1, 1121.1, 1162.6, 1142.6, 1077.9, 1027.3, 1078.2, 1094.3,
     G 969.83, 853.72, 849.91, 909.12, 995.68, 1095.0, 1146.9, 1086.3,
     H 1010.4, 1065.4, 1128.9, 1080.6, 987.93, 898.18, 835.20, 771.63,
     I 687.12, 614.52, 606.14, 737.09, 908.13, 997.64, 1080.6, 1126.3,
     J 1056.7, 1028.4, 1141.7, 1252.6, 1225.3, 1103.2, 1038.6, 1043.4,
     K 1002.9, 965.51, 1035.0, 1150.7, 1200.9, 1152.0, 1068.5, 995.84,
     L 889.52, 818.48, 907.01, 1042.2, 1055.6, 1000.6, 972.00, 985.72,
     M 1027.2, 1054.8, 1078.0, 1126.6, 1205.3, 1245.7, 1201.0, 1144.7,
     N 1097.5, 1030.1, 926.85, 836.71, 864.11, 993.50, 1075.3, 1032.6,
     O 1008.9, 1066.1, 1067.4, 1004.8, 971.54, 923.18, 815.71, 799.70,
     P 946.19, 1100.1, 1126.4, 1032.2, 895.14, 784.30, 734.77, 726.53,
     Q 726.88, 765.54, 863.90, 992.24, 1070.9, 1028.1, 858.78, 647.15,
     R 563.18, 679.98, 906.40, 1094.3, 1155.3, 1124.3, 1098.4, 1109.5 /
C         SOLAR SPECTRUM FROM  29840 TO  31270 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB02  /
     A 1076.2, 944.17, 849.20, 928.54, 1062.0, 1118.9, 1119.2, 1074.6,
     B 1005.8, 980.02, 999.11, 1002.4, 939.78, 838.12, 816.13, 908.73,
     C 1014.9, 1058.3, 1043.7, 987.54, 946.35, 981.40, 1055.8, 1094.3,
     D 1028.3, 916.41, 908.99, 991.83, 1049.6, 1076.2, 1093.5, 1076.3,
     E 1014.5, 949.61, 947.26, 1001.2, 1051.5, 1072.8, 1068.0, 1012.5,
     F 907.81, 866.30, 950.89, 1037.5, 1079.5, 1183.9, 1291.3, 1268.6,
     G 1199.3, 1188.6, 1188.0, 1186.6, 1198.2, 1171.3, 1132.6, 1131.6,
     H 1096.0, 971.10, 847.07, 836.62, 922.78, 990.99, 987.51, 969.24,
     I 981.46, 981.36, 971.95, 985.34, 1003.0, 1037.2, 1071.2, 1065.7,
     J 1026.7, 984.84, 1002.7, 1070.3, 1117.5, 1116.0, 1048.9, 965.34,
     K 972.27, 1045.7, 1096.6, 1127.5, 1133.5, 1099.6, 1079.3, 1082.9,
     L 1026.8, 927.50, 879.08, 858.83, 831.01, 807.82, 789.56, 813.75,
     M 893.46, 937.62, 901.56, 864.46, 873.35, 891.03, 862.46, 810.30,
     N 787.36, 752.93, 715.34, 708.07, 728.93, 786.79, 807.73, 736.28,
     O 645.08, 616.90, 649.17, 691.77, 749.18, 820.21, 820.68, 791.26,
     P 854.27, 940.56, 956.38, 909.42, 824.18, 767.17, 722.06, 653.42,
     Q 624.67, 633.73, 655.14, 707.93, 784.94, 880.79, 961.15, 985.60,
     R 986.18, 966.53, 921.47, 888.89, 855.85, 851.66, 886.78, 850.97 /
C         SOLAR SPECTRUM FROM  31280 TO  32710 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB03  /
     A 766.97, 738.95, 724.53, 657.61, 587.77, 616.86, 760.61, 903.23,
     B 917.27, 838.49, 784.80, 759.41, 719.61, 671.48, 624.63, 588.57,
     C 574.70, 596.68, 698.02, 866.39, 974.82, 960.37, 930.10, 962.65,
     D 1007.1, 1001.9, 926.29, 816.64, 763.25, 772.93, 762.66, 729.39,
     E 725.01, 727.16, 672.73, 581.42, 520.97, 488.80, 478.60, 542.08,
     F 663.71, 749.48, 785.87, 811.05, 818.19, 813.80, 824.54, 836.62,
     G 799.66, 728.00, 660.36, 559.28, 473.28, 550.16, 752.04, 885.84,
     H 906.80, 912.21, 929.32, 899.72, 830.20, 774.56, 736.42, 724.09,
     I 740.12, 754.11, 764.96, 780.76, 788.94, 784.87, 758.80, 725.91,
     J 751.84, 804.24, 777.73, 703.36, 665.27, 663.99, 679.36, 706.09,
     K 757.57, 836.09, 880.02, 881.18, 907.91, 929.26, 894.32, 874.01,
     L 918.56, 953.50, 922.32, 866.61, 836.54, 825.28, 752.54, 586.02,
```

```
     M 427.46, 374.05, 437.23, 534.32, 556.74, 563.11, 629.31, 631.26,
     N 518.76, 438.31, 460.31, 530.45, 608.50, 657.99, 662.08, 686.17,
     O 775.18, 843.11, 797.46, 685.33, 611.33, 628.74, 711.36, 754.94,
     P 728.80, 722.79, 726.38, 679.68, 665.83, 710.48, 723.10, 724.09,
     Q 760.18, 784.01, 742.78, 634.33, 546.55, 563.54, 611.03, 623.16,
     R 665.36, 743.55, 764.46, 671.14, 513.18, 401.86, 405.77, 515.72 /
C         SOLAR SPECTRUM FROM  32720 TO  34150 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB04  /
     A 639.90, 677.85, 679.55, 759.33, 848.11, 819.89, 751.75, 710.50,
     B 615.33, 525.09, 583.35, 715.23, 767.53, 739.10, 664.05, 580.57,
     C 572.85, 634.13, 648.77, 561.27, 497.72, 591.71, 737.83, 794.19,
     D 802.51, 799.33, 735.79, 658.41, 659.47, 718.18, 761.67, 697.24,
     E 545.14, 474.47, 526.96, 597.65, 584.74, 447.28, 291.35, 261.28,
     F 330.26, 401.96, 466.32, 531.26, 572.34, 584.86, 585.17, 569.46,
     G 558.27, 559.41, 512.02, 426.37, 378.14, 398.26, 473.49, 542.18,
     H 531.76, 437.48, 341.85, 305.82, 299.88, 328.12, 440.04, 586.46,
     I 660.32, 625.22, 510.26, 418.85, 447.36, 534.89, 605.86, 667.07,
     J 687.31, 636.79, 549.63, 472.88, 419.53, 370.06, 327.98, 320.49,
     K 354.00, 399.17, 450.98, 528.34, 608.25, 696.07, 774.28, 760.75,
     L 690.58, 648.20, 580.63, 477.96, 453.91, 488.74, 464.02, 421.59,
     M 444.32, 446.59, 375.95, 342.13, 397.49, 510.97, 646.38, 725.14,
     N 703.06, 639.06, 619.10, 654.66, 665.99, 611.40, 580.22, 607.29,
     O 591.05, 542.30, 583.82, 673.02, 673.21, 582.44, 465.73, 377.25,
     P 377.04, 487.27, 607.93, 617.52, 583.46, 601.68, 615.94, 575.47,
     Q 541.63, 542.06, 522.28, 472.49, 423.29, 438.09, 556.72, 664.34,
     R 669.88, 657.45, 684.71, 705.70, 683.11, 600.81, 509.90, 497.64 /
C         SOLAR SPECTRUM FROM  34160 TO  35590 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB05  /
     A 511.07, 496.07, 500.32, 518.70, 529.91, 563.00, 609.20, 626.49,
     B 622.11, 615.72, 600.44, 591.26, 598.12, 593.07, 590.94, 631.58,
     C 696.48, 718.48, 676.11, 631.56, 619.64, 620.53, 624.10, 636.56,
     D 658.02, 688.78, 724.81, 742.60, 722.31, 675.86, 665.96, 704.73,
     E 703.70, 645.00, 598.26, 587.77, 590.94, 575.93, 528.03, 477.92,
     F 457.52, 456.80, 454.91, 448.65, 445.47, 445.38, 444.43, 446.04,
     G 455.91, 468.02, 454.34, 393.32, 301.22, 211.44, 167.11, 193.99,
     H 254.01, 305.35, 353.03, 385.08, 387.03, 391.60, 406.20, 415.34,
     I 435.34, 469.77, 492.15, 472.73, 409.86, 353.25, 340.68, 355.27,
     J 379.77, 401.81, 409.67, 406.89, 393.16, 378.89, 375.20, 373.52,
     K 360.19, 322.79, 273.55, 237.76, 212.33, 184.80, 156.20, 127.75,
     L 96.269, 68.806, 62.047, 77.143, 100.47, 127.56, 159.88, 194.05,
     M 225.20, 254.64, 285.75, 300.14, 294.40, 308.92, 340.83, 346.26,
     N 336.29, 347.54, 373.81, 388.78, 372.68, 325.29, 294.40, 317.56,
     O 360.30, 378.08, 374.22, 374.03, 383.34, 387.88, 377.55, 356.96,
     P 340.67, 328.71, 314.00, 316.91, 344.51, 355.54, 335.66, 318.68,
     Q 318.65, 322.43, 318.61, 304.92, 284.84, 268.13, 265.80, 273.55,
     R 274.18, 252.38, 215.04, 188.60, 181.31, 181.31, 180.78, 175.24 /
C         SOLAR SPECTRUM FROM  35600 TO  37030 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB06  /
     A 162.06, 145.08, 128.76, 113.76, 98.078, 83.072, 76.222, 78.359,
     B 78.434, 74.235, 75.843, 80.321, 77.859, 70.298, 64.651, 67.049,
     C 77.810, 83.167, 75.286, 71.202, 80.549, 92.008, 100.17, 108.63,
     D 119.44, 130.78, 142.31, 158.94, 177.12, 186.40, 186.60, 181.47,
     E 175.30, 175.54, 179.00, 177.04, 172.60, 172.67, 178.98, 193.77,
     F 215.13, 233.62, 252.05, 277.68, 298.91, 298.40, 280.81, 274.21,
     G 286.52, 285.46, 259.71, 241.39, 246.98, 259.87, 274.27, 298.47,
     H 316.85, 303.19, 263.69, 229.31, 227.90, 256.12, 281.58, 300.19,
     I 310.56, 279.54, 211.93, 152.18, 129.94, 147.47, 181.62, 215.37,
     J 239.50, 233.12, 191.55, 139.41, 110.51, 118.93, 134.79, 129.05,
     K 124.39, 143.53, 158.29, 141.84, 116.32, 111.59, 128.93, 149.17,
     L 153.44, 145.63, 148.52, 159.25, 155.84, 154.17, 177.28, 203.40,
     M 207.35, 205.27, 222.85, 253.18, 271.28, 279.27, 302.17, 321.47,
     N 288.83, 230.14, 206.40, 213.22, 216.49, 207.46, 196.20, 195.21,
     O 202.03, 194.33, 164.86, 136.65, 123.87, 128.14, 161.89, 216.99,
     P 253.68, 249.26, 222.89, 213.11, 243.64, 293.10, 309.42, 286.40,
     Q 269.61, 272.23, 271.67, 265.84, 265.61, 264.77, 266.03, 289.51,
     R 325.67, 337.34, 321.17, 300.30, 282.60, 287.14, 322.06, 335.79 /
C         SOLAR SPECTRUM FROM  37040 TO  38470 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB07  /
     A 297.22, 254.10, 243.47, 239.49, 219.32, 211.94, 239.28, 271.43,
     B 279.37, 272.26, 264.77, 250.52, 229.93, 222.15, 235.30, 256.79,
     C 275.28, 286.92, 284.85, 269.52, 255.05, 253.46, 263.22, 274.78,
     D 279.19, 270.17, 249.41, 229.04, 221.64, 231.38, 252.70, 280.64,
     E 310.06, 328.33, 325.01, 290.26, 238.97, 223.38, 257.24, 282.60,
     F 264.32, 243.34, 253.18, 272.89, 271.32, 256.12, 260.24, 271.35,
     G 257.11, 236.61, 238.72, 248.92, 255.90, 272.04, 291.78, 297.40,
     H 288.09, 283.28, 292.92, 301.74, 309.07, 322.05, 320.42, 295.43,
     I 269.65, 254.41, 240.88, 228.18, 221.23, 213.72, 201.23, 197.17,
     J 212.29, 233.39, 247.65, 261.74, 286.17, 322.49, 349.47, 338.28,
     K 297.06, 261.55, 252.28, 264.65, 286.92, 298.94, 280.45, 244.37,
     L 213.47, 193.03, 182.07, 168.54, 143.12, 114.10, 89.615, 73.589,
     M 73.990, 87.912, 96.265, 94.813, 96.604, 102.30, 102.15, 103.07,
     N 117.81, 137.41, 146.09, 144.28, 137.89, 128.11, 122.82, 128.19,
     O 130.66, 117.31, 98.912, 93.397, 105.63, 122.73, 126.39, 113.05,
     P 92.317, 76.340, 69.032, 66.324, 71.280, 87.431, 105.94, 114.02,
     Q 107.91, 91.872, 75.208, 69.123, 75.930, 90.928, 109.71, 125.70,
     R 135.79, 141.14, 138.14, 121.33, 91.806, 63.497, 52.106, 59.555 /
C         SOLAR SPECTRUM FROM  38480 TO  39910 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB08  /
     A 81.015, 106.67, 118.97, 116.36, 110.82, 100.88, 89.056, 90.431,
     B 104.41, 114.95, 124.85, 148.87, 171.72, 167.22, 142.25, 118.42,
     C 98.653, 78.908, 68.133, 77.286, 100.93, 120.08, 125.49, 131.79,
     D 155.69, 180.75, 181.81, 166.77, 150.06, 133.24, 116.14, 97.728,
     E 81.629, 76.695, 87.607, 110.23, 134.88, 149.13, 147.64, 139.88,
     F 135.19, 135.07, 138.00, 136.73, 128.84, 122.22, 120.48, 121.98,
     G 123.08, 116.30, 101.43, 86.303, 74.719, 68.800, 71.327, 80.626,
     H 90.485, 96.739, 100.69, 100.81, 93.677, 84.740, 81.532, 82.893,
     I 84.564, 87.584, 91.780, 91.272, 87.014, 87.386, 90.149, 84.917,
     J 71.266, 57.873, 51.863, 53.876, 57.909, 58.508, 57.020, 57.432,
     K 60.671, 64.667, 67.362, 67.511, 64.233, 59.035, 55.697, 56.636,
     L 59.400, 59.070, 56.522, 55.834, 55.860, 54.039, 51.976, 52.344,
     M 54.667, 56.450, 56.751, 56.769, 58.002, 60.029, 59.602, 53.134,
     N 42.926, 35.588, 33.447, 35.171, 39.379, 44.371, 47.745, 46.933,
     O 42.441, 37.879, 35.595, 36.458, 41.048, 47.300, 51.098, 50.024,
     P 45.331, 41.282, 40.082, 40.000, 39.104, 37.329, 36.632, 37.792,
     Q 39.189, 41.058, 45.214, 50.737, 54.281, 55.015, 56.138, 60.931,
     R 67.383, 69.534, 65.159, 56.372, 47.326, 44.322, 49.944, 59.696 /
C         SOLAR SPECTRUM FROM  39920 TO  41350 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB09  /
     A 67.929, 71.334, 69.905, 65.620, 59.303, 54.016, 55.880, 65.155,
     B 74.065, 76.217, 73.506, 71.406, 70.849, 69.749, 69.268, 71.380,
     C 72.721, 68.929, 61.665, 54.896, 47.420, 38.325, 32.219, 31.243,
     D 33.310, 35.358, 35.623, 36.840, 41.551, 47.499, 51.176, 50.344,
     E 45.362, 38.341, 33.130, 33.801, 40.140, 49.121, 55.385, 55.174,
     F 50.450, 46.511, 47.495, 51.883, 56.354, 59.603, 61.584, 63.215,
     G 64.603, 64.101, 59.027, 50.956, 47.633, 52.543, 58.883, 59.829,
     H 57.617, 56.727, 57.371, 57.898, 57.177, 55.129, 52.952, 52.018,
     I 52.186, 52.044, 50.269, 46.592, 42.515, 40.755, 41.887, 44.119,
     J 46.536, 48.858, 50.490, 51.919, 54.085, 54.707, 51.927, 49.449,
     K 49.865, 50.933, 50.496, 48.616, 46.717, 46.070, 46.263, 46.733,
     L 48.009, 50.187, 52.420, 53.536, 52.507, 51.380, 53.214, 56.985,
     M 60.614, 63.139, 63.999, 63.869, 65.100, 69.385, 74.743, 78.184,
     N 78.103, 74.113, 67.371, 60.849, 58.924, 62.682, 68.032, 69.117,
     O 64.604, 59.110, 55.998, 56.838, 61.778, 65.874, 65.079, 63.038,
     P 64.809, 69.911, 74.841, 76.439, 73.587, 68.853, 67.497, 72.675,
     Q 80.602, 83.422, 78.957, 72.228, 66.737, 62.842, 61.535, 63.574,
     R 69.248, 76.577, 79.922, 77.755, 73.938, 70.518, 68.003, 66.339 /
C         SOLAR SPECTRUM FROM  41360 TO  42790 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB10  /
     A 63.979, 61.098, 59.421, 58.103, 55.741, 52.549, 48.079, 42.578,
     B 38.373, 37.297, 37.455, 34.861, 30.483, 29.634, 34.734, 42.460,
```

```
     C 47.066, 45.848, 40.157, 34.290, 31.584, 30.650, 29.054, 27.788,
     D 30.427, 37.570, 44.196, 46.880, 47.848, 49.166, 49.180, 45.002,
     E 38.135, 35.055, 38.095, 41.750, 40.899, 35.722, 28.884, 24.835,
     F 28.670, 39.646, 50.310, 55.725, 57.401, 58.110, 59.406, 59.360,
     G 53.420, 43.004, 34.787, 33.697, 39.682, 47.554, 52.605, 53.632,
     H 51.001, 45.266, 37.844, 31.030, 25.936, 22.799, 21.882, 23.484,
     I 27.857, 33.447, 37.319, 39.195, 42.826, 50.398, 58.752, 63.301,
     J 61.094, 53.532, 46.046, 41.118, 37.646, 36.304, 40.426, 50.893,
     K 61.553, 65.395, 62.680, 58.087, 54.622, 51.330, 46.874, 42.870,
     L 40.547, 39.760, 40.217, 40.359, 39.559, 40.667, 46.260, 53.413,
     M 56.041, 52.566, 46.674, 41.073, 35.511, 31.231, 31.082, 35.955,
     N 45.199, 55.464, 61.802, 63.505, 61.850, 56.412, 49.388, 46.369,
     O 50.058, 56.694, 60.884, 61.030, 58.107, 54.303, 51.940, 50.508,
     P 46.749, 39.155, 31.535, 28.959, 30.973, 32.670, 31.567, 29.340,
     Q 27.275, 25.184, 24.264, 27.068, 34.296, 42.475, 47.230, 47.425,
     R 44.435, 40.538, 36.868, 33.020, 29.405, 28.753, 34.079, 44.246 /
C         SOLAR SPECTRUM FROM  42800 TO  44230 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB11  /
     A 53.780, 57.974, 56.376, 51.200, 45.308, 40.273, 35.900, 33.344,
     B 34.011, 36.858, 41.283, 47.374, 53.088, 56.201, 55.633, 50.843,
     C 43.997, 38.767, 36.248, 36.380, 40.762, 50.700, 63.371, 73.432,
     D 76.418, 70.373, 58.741, 47.034, 38.598, 34.664, 35.794, 42.084,
     E 49.973, 54.338, 53.956, 52.287, 52.778, 55.571, 59.034, 60.268,
     F 56.247, 47.362, 38.056, 32.889, 31.739, 31.734, 32.476, 35.060,
     G 39.091, 43.398, 48.131, 53.574, 58.749, 63.599, 68.971, 73.421,
     H 73.861, 69.003, 60.557, 51.865, 44.879, 42.060, 44.802, 47.950,
     I 46.882, 42.973, 39.293, 37.711, 37.137, 35.222, 32.243, 30.488,
     J 32.605, 40.429, 51.099, 57.710, 57.150, 52.992, 50.275, 49.986,
     K 49.778, 48.371, 46.421, 44.604, 42.730, 41.244, 41.565, 43.805,
     L 47.013, 48.992, 46.428, 40.595, 37.840, 42.353, 52.248, 60.529,
     M 61.566, 56.800, 52.041, 52.260, 57.077, 61.019, 60.712, 57.048,
     N 51.481, 46.352, 44.366, 44.947, 45.478, 44.944, 43.825, 42.105,
     O 39.466, 36.826, 35.907, 36.357, 35.661, 33.947, 33.690, 34.429,
     P 34.000, 32.645, 31.410, 30.281, 29.409, 29.127, 29.326, 29.869,
     Q 30.601, 31.311, 32.099, 32.779, 32.757, 32.098, 31.975, 33.484,
     R 36.048, 39.169, 43.365, 47.244, 48.214, 45.786, 41.586, 38.775 /
C         SOLAR SPECTRUM FROM  44240 TO  45670 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB12  /
     A 40.753, 46.752, 51.684, 52.597, 51.449, 50.684, 49.450, 46.747,
     B 45.369, 47.685, 50.240, 48.961, 46.693, 48.600, 53.694, 56.465,
     C 54.341, 50.722, 49.877, 51.246, 52.088, 52.765, 56.254, 63.326,
     D 69.744, 71.066, 68.349, 65.123, 62.551, 59.195, 53.705, 48.161,
     E 46.236, 47.710, 49.660, 50.799, 51.836, 54.537, 59.647, 64.707,
     F 65.844, 61.634, 55.570, 54.083, 58.781, 64.888, 69.777, 74.008,
     G 76.492, 76.226, 74.746, 74.941, 77.801, 79.619, 76.190, 67.190,
     H 55.231, 45.813, 43.141, 45.647, 49.466, 52.231, 52.221, 48.886,
     I 44.716, 42.613, 43.385, 45.968, 48.121, 48.998, 49.885, 50.707,
     J 49.893, 48.319, 48.198, 50.280, 53.830, 55.914, 54.822, 52.939,
     K 51.944, 49.438, 42.956, 34.614, 28.100, 24.503, 24.203, 27.839,
     L 34.604, 41.615, 45.324, 45.444, 45.527, 47.179, 45.756, 36.862,
     M 26.037, 20.569, 20.329, 24.263, 30.863, 35.939, 36.711, 35.693,
     N 37.256, 40.862, 44.416, 48.800, 54.182, 57.655, 58.427, 59.965,
     O 63.940, 66.820, 65.465, 59.482, 49.396, 39.422, 34.182, 35.388,
     P 42.875, 52.034, 57.595, 59.093, 57.272, 52.172, 45.493, 39.419,
     Q 35.581, 35.902, 40.354, 46.732, 53.309, 58.781, 61.785, 59.255,
     R 50.030, 41.567, 40.523, 43.584, 44.875, 42.754, 40.077, 39.941 /
C         SOLAR SPECTRUM FROM  45680 TO  47110 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB13  /
     A 40.977, 39.567, 34.955, 30.424, 31.039, 38.687, 47.480, 49.830,
     B 46.790, 44.829, 46.546, 50.415, 54.602, 57.656, 58.463, 57.276,
     C 55.621, 54.514, 53.338, 50.026, 42.817, 33.636, 27.134, 25.516,
     D 27.897, 31.392, 32.125, 29.463, 26.581, 25.956, 27.737, 31.175,
     E 34.959, 37.671, 38.641, 37.958, 36.733, 35.681, 33.877, 30.849,
     F 28.059, 27.615, 29.319, 29.375, 25.390, 20.659, 19.484, 22.297,
     G 27.282, 32.467, 35.906, 37.137, 37.895, 39.130, 39.777, 39.872,
     H 40.778, 42.317, 42.934, 40.430, 34.227, 27.701, 23.880, 22.174,
     I 21.639, 22.589, 25.184, 29.017, 32.981, 36.110, 38.580, 41.239,
     J 44.426, 46.939, 47.010, 44.165, 39.659, 35.556, 32.838, 31.546,
     K 32.676, 36.963, 42.333, 44.931, 43.704, 40.943, 37.973, 35.199,
     L 33.574, 33.339, 34.185, 36.347, 39.963, 43.964, 47.162, 48.987,
     M 48.976, 47.948, 48.004, 49.892, 51.065, 47.834, 40.489, 32.665,
     N 26.795, 24.461, 26.655, 31.928, 37.634, 41.345, 40.956, 36.827,
     O 32.110, 28.612, 26.482, 26.602, 28.831, 30.877, 30.976, 30.063,
     P 29.887, 30.305, 29.974, 28.265, 26.517, 27.066, 30.403, 34.539,
     Q 37.104, 37.598, 37.252, 37.060, 36.498, 34.167, 29.814, 24.192,
     R 18.515, 15.086, 15.040, 17.158, 20.807, 25.682, 30.352, 34.203 /
C         SOLAR SPECTRUM FROM  47120 TO  48550 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB14  /
     A 37.902, 42.531, 47.832, 50.509, 48.019, 42.616, 38.321, 37.370,
     B 40.172, 44.395, 46.132, 43.911, 38.396, 31.379, 26.275, 25.075,
     C 26.652, 28.963, 31.168, 34.168, 38.050, 40.231, 38.347, 32.741,
     D 26.199, 21.863, 20.249, 20.185, 21.726, 25.562, 30.318, 33.431,
     E 34.453, 34.959, 36.374, 37.870, 36.655, 31.966, 25.920, 21.264,
     F 20.663, 24.658, 30.263, 34.021, 34.336, 31.356, 26.926, 23.109,
     G 20.867, 20.684, 22.416, 24.878, 26.779, 27.334, 26.537, 25.210,
     H 24.013, 22.944, 21.800, 20.449, 19.290, 19.528, 21.742, 24.125,
     I 23.994, 21.559, 19.555, 18.915, 18.342, 17.335, 16.549, 16.479,
     J 17.211, 18.445, 19.294, 18.980, 17.912, 17.156, 17.103, 17.256,
     K 16.925, 15.842, 14.485, 13.683, 13.647, 13.914, 14.009, 13.770,
     L 13.456, 13.399, 13.547, 13.760, 14.060, 14.427, 14.644, 14.438,
     M 13.986, 13.749, 13.927, 14.390, 14.759, 14.822, 14.679, 14.448,
     N 14.186, 13.937, 13.754, 13.657, 13.540, 13.308, 13.053, 12.841,
     O 12.704, 12.742, 12.811, 12.662, 12.355, 12.100, 12.003, 12.014,
     P 12.067, 12.223, 12.444, 12.472, 12.164, 11.732, 11.515, 11.619,
     Q 11.873, 12.028, 11.947, 11.722, 11.399, 10.930, 10.473, 10.205,
     R 10.224, 10.694, 11.468, 12.007, 12.083, 11.905, 11.498, 10.891 /
C         SOLAR SPECTRUM FROM  48560 TO  49990 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB15  /
     A 10.575, 10.846, 11.353, 11.612, 11.411, 10.876, 10.383, 10.305,
     B 10.695, 11.245, 11.636, 11.828, 11.918, 11.865, 11.674, 11.510,
     C 11.407, 11.303, 11.216, 11.143, 11.039, 10.983, 11.004, 10.900,
     D 10.653, 10.562, 10.781, 11.186, 11.605, 11.806, 11.582, 11.056,
     E 10.567, 10.335, 10.408, 10.729, 11.165, 11.540, 11.646, 11.372,
     F 10.933, 10.524, 9.9973, 9.3783, 8.9883, 9.0163, 9.4125, 9.9179,
     G 10.278, 10.472, 10.553, 10.575, 10.519, 10.216, 9.6821, 9.1499,
     H 8.7057, 8.3894, 8.3442, 8.6241, 9.1371, 9.7184, 10.191, 10.443,
     I 10.458, 10.289, 9.9772, 9.5829, 9.3097, 9.3195, 9.4694, 9.5182,
     J 9.4326, 9.2478, 8.8197, 7.9809, 6.9996, 6.4856, 6.7462, 7.5406,
     K 8.2813, 8.7258, 9.0682, 9.1665, 8.8637, 8.4638, 8.2393, 8.1656,
     L 8.1880, 8.3578, 8.6488, 8.8980, 9.0117, 9.0659, 9.1955, 9.4207,
     M 9.5526, 9.4237, 9.1290, 8.8441, 8.6138, 8.4237, 8.2979, 8.2598,
     N 8.2859, 8.3475, 8.4533, 8.6285, 8.8310, 8.8866, 8.6750, 8.3312,
     O 8.0091, 7.7296, 7.6239, 7.8692, 8.2725, 8.4086, 8.2515, 8.0914,
     P 8.0003, 7.9367, 7.9266, 7.9580, 8.0492, 8.2376, 8.4263, 8.4811,
     Q 8.3309, 8.0263, 7.7632, 7.6987, 7.8124, 7.9390, 8.0183, 8.0816,
     R 8.0428, 7.8923, 7.6963, 7.4969, 7.4013, 7.4289, 7.4489, 7.4059 /
C         SOLAR SPECTRUM FROM  50000 TO  51430 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB16  /
     A 7.4198, 7.5261, 7.5252, 7.3239, 7.1263, 7.1423, 7.3340, 7.5049,
     B 7.5484, 7.5319, 7.5163, 7.4995, 7.5728, 7.8104, 8.0588, 8.0948,
     C 7.9140, 7.6978, 7.5116, 7.2138, 6.8063, 6.5430, 6.5232, 6.5869,
     D 6.5610, 6.3984, 6.1889, 6.0587, 6.0676, 6.1988, 6.3140, 6.2527,
     E 6.0929, 6.0277, 6.0941, 6.3031, 6.6594, 6.9398, 6.9566, 6.8310,
     F 6.7374, 6.6812, 6.6558, 6.8336, 7.2020, 7.4012, 7.2950, 7.0488,
     G 6.7966, 6.6293, 6.5868, 6.5980, 6.6007, 6.6501, 6.7627, 6.7853,
     H 6.6321, 6.4856, 6.5198, 6.6486, 6.7271, 6.7227, 6.6696, 6.6189,
     I 6.5979, 6.6188, 6.7110, 6.8343, 6.8750, 6.8250, 6.7885, 6.8266,
     J 6.8556, 6.8068, 6.8377, 7.0467, 7.2779, 7.4139, 7.4712, 7.4621,
     K 7.4071, 7.3592, 7.3372, 7.3220, 7.2938, 7.2531, 7.2052, 7.1335,
     L 7.0298, 6.8533, 6.5535, 6.2227, 6.0139, 5.9384, 5.9038, 5.8568,
```

```
     M 5.7909, 5.7326, 5.7745, 5.9608, 6.1865, 6.3681, 6.4997, 6.5437,
     N 6.4637, 6.2708, 6.0451, 5.9557, 6.0855, 6.2542, 6.2454, 6.0795,
     O 5.9102, 5.8447, 5.9218, 6.1063, 6.2895, 6.3271, 6.1097, 5.7421,
     P 5.4452, 5.2981, 5.3256, 5.4935, 5.6819, 5.8245, 5.8933, 5.9630,
     Q 6.1703, 6.4525, 6.6325, 6.6965, 6.7185, 6.6238, 6.3107, 5.9241,
     R 5.6987, 5.6651, 5.7428, 5.8790, 5.9715, 5.9618, 5.9674, 6.0754 /
C          SOLAR SPECTRUM FROM  51440 TO  52870 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB17  /
     A 6.2541, 6.4300, 6.4968, 6.4564, 6.4082, 6.3024, 6.0135, 5.6431,
     B 5.3963, 5.2989, 5.2635, 5.2227, 5.1279, 4.9315, 4.6348, 4.3168,
     C 4.0151, 3.6625, 3.2906, 3.1028, 3.1349, 3.1994, 3.2596, 3.4144,
     D 3.5949, 3.6534, 3.6296, 3.6281, 3.5876, 3.4292, 3.2659, 3.2284,
     E 3.2576, 3.3002, 3.4535, 3.7372, 4.0573, 4.3558, 4.5999, 4.7781,
     F 4.8855, 4.8999, 4.8392, 4.7624, 4.7059, 4.6981, 4.7666, 4.8453,
     G 4.8236, 4.7293, 4.6861, 4.7132, 4.7725, 4.8713, 4.9596, 4.9527,
     H 4.8957, 4.9252, 5.0736, 5.2229, 5.2505, 5.1537, 5.0156, 4.8880,
     I 4.7686, 4.6549, 4.5534, 4.4828, 4.4661, 4.5040, 4.5905, 4.7033,
     J 4.7858, 4.8334, 4.9283, 5.0377, 5.0065, 4.8471, 4.6828, 4.5586,
     K 4.4812, 4.4314, 4.3903, 4.3830, 4.4066, 4.3900, 4.2973, 4.1978,
     L 4.1462, 4.1084, 4.1495, 4.3897, 4.6859, 4.8206, 4.7938, 4.6781,
     M 4.5222, 4.3959, 4.3358, 4.2947, 4.2259, 4.1452, 4.1060, 4.1462,
     N 4.2149, 4.2549, 4.3061, 4.3742, 4.3738, 4.2718, 4.1389, 4.0405,
     O 3.9457, 3.8127, 3.7099, 3.7344, 3.8589, 3.9598, 3.9525, 3.8377,
     P 3.6708, 3.5357, 3.4929, 3.5375, 3.6381, 3.7890, 3.9671, 4.0995,
     Q 4.1421, 4.1302, 4.1235, 4.1623, 4.2506, 4.2948, 4.2231, 4.0993,
     R 3.9680, 3.9475, 4.1958, 4.5131, 4.6101, 4.5130, 4.3474, 4.1749 /
C          SOLAR SPECTRUM FROM  52880 TO  54310 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB18  /
     A 4.0467, 3.9956, 4.0078, 4.0374, 4.0255, 3.9379, 3.8192, 3.7529,
     B 3.7675, 3.8260, 3.8654, 3.8518, 3.8148, 3.8028, 3.8098, 3.7934,
     C 3.7660, 3.7944, 3.8689, 3.8978, 3.8856, 3.8923, 3.8570, 3.6940,
     D 3.4693, 3.3222, 3.2824, 3.2887, 3.3039, 3.3222, 3.3313, 3.3326,
     E 3.3482, 3.3807, 3.4188, 3.4602, 3.4972, 3.5151, 3.5155, 3.5165,
     F 3.5258, 3.5406, 3.5478, 3.5345, 3.5339, 3.5820, 3.6396, 3.6448,
     G 3.5872, 3.5112, 3.4804, 3.5257, 3.6238, 3.7290, 3.8023, 3.8024,
     H 3.7268, 3.6578, 3.6439, 3.6422, 3.6373, 3.6397, 3.6410, 3.6494,
     I 3.6608, 3.6251, 3.5212, 3.4020, 3.2845, 3.1230, 2.9483, 2.8515,
     J 2.8432, 2.8638, 2.8967, 2.9505, 3.0025, 3.0552, 3.1106, 3.1178,
     K 3.0596, 2.9854, 2.9316, 2.8903, 2.8590, 2.8500, 2.8450, 2.8121,
     L 2.7626, 2.7424, 2.7667, 2.8024, 2.8165, 2.8111, 2.8128, 2.8569,
     M 2.9659, 3.1062, 3.1990, 3.2128, 3.2088, 3.2391, 3.2661, 3.2364,
     N 3.1173, 2.9094, 2.6952, 2.5324, 2.3959, 2.2953, 2.2510, 2.2245,
     O 2.1811, 2.1301, 2.1482, 2.3257, 2.5856, 2.7226, 2.6495, 2.4508,
     P 2.2444, 2.0850, 1.9891, 1.9843, 2.0816, 2.2233, 2.3248, 2.3551,
     Q 2.3479, 2.3606, 2.4296, 2.5361, 2.6128, 2.6216, 2.6069, 2.6196,
     R 2.6464, 2.6427, 2.5823, 2.4682, 2.3320, 2.2405, 2.2637, 2.3973 /
C          SOLAR SPECTRUM FROM  54320 TO  55750 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB19  /
     A 2.5524, 2.6891, 2.8508, 3.0103, 3.0681, 3.0064, 2.9114, 2.8609,
     B 2.8517, 2.8374, 2.7894, 2.7288, 2.7138, 2.7729, 2.8707, 2.9536,
     C 2.9953, 2.9911, 2.9398, 2.8550, 2.7732, 2.7303, 2.7366, 2.7650,
     D 2.7705, 2.7374, 2.6830, 2.6218, 2.5663, 2.5341, 2.5351, 2.5681,
     E 2.6124, 2.6305, 2.6024, 2.5431, 2.4840, 2.4546, 2.4684, 2.5100,
     F 2.5445, 2.5532, 2.5564, 2.5889, 2.6616, 2.7553, 2.8466, 2.9290,
     G 2.9958, 3.0175, 2.9774, 2.8990, 2.8001, 2.6927, 2.6171, 2.5931,
     H 2.5809, 2.5276, 2.4284, 2.3365, 2.3162, 2.3855, 2.4872, 2.5455,
     I 2.5773, 2.6809, 2.9720, 3.5757, 4.4006, 5.0044, 5.0295, 4.5135,
     J 3.7071, 2.9059, 2.3600, 2.1418, 2.1119, 2.0871, 2.0301, 2.0043,
     K 2.0361, 2.0963, 2.1520, 2.1878, 2.1955, 2.1864, 2.1899, 2.2170,
     L 2.2574, 2.2895, 2.2783, 2.2148, 2.1641, 2.2343, 2.4726, 2.8119,
     M 3.1288, 3.2984, 3.2206, 2.8859, 2.4473, 2.1436, 2.0729, 2.1391,
     N 2.2171, 2.2580, 2.2654, 2.2481, 2.2103, 2.1657, 2.1356, 2.1321,
     O 2.1438, 2.1461, 2.1396, 2.1460, 2.1588, 2.1581, 2.1481, 2.1343,
     P 2.1101, 2.0754, 2.0400, 2.0121, 1.9930, 1.9799, 1.9699, 1.9613,
     Q 1.9537, 1.9454, 1.9312, 1.9058, 1.8726, 1.8470, 1.8465, 1.8693,
     R 1.8844, 1.8635, 1.8143, 1.7618, 1.7188, 1.6853, 1.6656, 1.6708 /
C          SOLAR SPECTRUM FROM  55760 TO  57190 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB20  /
     A 1.7036, 1.7519, 1.8120, 1.9015, 2.0124, 2.0980, 2.1385, 2.1481,
     B 2.1347, 2.1086, 2.0953, 2.1062, 2.1095, 2.0685, 2.0001, 1.9461,
     C 1.9194, 1.9088, 1.9023, 1.8977, 1.9049, 1.9300, 1.9588, 1.9635,
     D 1.9357, 1.9019, 1.8887, 1.8939, 1.9018, 1.9038, 1.8975, 1.8747,
     E 1.8289, 1.7716, 1.7303, 1.7330, 1.7900, 1.8782, 1.9548, 1.9907,
     F 1.9807, 1.9430, 1.9173, 1.9218, 1.9203, 1.8717, 1.7832, 1.6965,
     G 1.6389, 1.6077, 1.5924, 1.5818, 1.5583, 1.5142, 1.4616, 1.4237,
     H 1.4252, 1.4834, 1.5970, 1.7410, 1.8771, 1.9784, 2.0451, 2.0872,
     I 2.0909, 2.0384, 1.9573, 1.9002, 1.8824, 1.8663, 1.8193, 1.7540,
     J 1.6874, 1.6222, 1.5726, 1.5450, 1.5290, 1.5312, 1.5699, 1.6411,
     K 1.7186, 1.7678, 1.7546, 1.6623, 1.5115, 1.3588, 1.2605, 1.2348,
     L 1.2611, 1.3091, 1.3588, 1.3884, 1.3800, 1.3482, 1.3224, 1.3159,
     M 1.3437, 1.4142, 1.4950, 1.5443, 1.5521, 1.5282, 1.4902, 1.4606,
     N 1.4465, 1.4398, 1.4399, 1.4544, 1.4760, 1.4781, 1.4506, 1.4229,
     O 1.4185, 1.4221, 1.4119, 1.3908, 1.3779, 1.3813, 1.3933, 1.4087,
     P 1.4268, 1.4417, 1.4408, 1.4188, 1.3861, 1.3548, 1.3261, 1.2980,
     Q 1.2769, 1.2731, 1.2856, 1.3002, 1.3056, 1.2987, 1.2817, 1.2590,
     R 1.2291, 1.1868, 1.1428, 1.1183, 1.1141, 1.1120, 1.1009, 1.0797 /
C     SOLAR SPECTRUM FROM  57200 TO  57490 CM-1, IN STEPS OF 10 CM-1
      DATA SUNB21  /
     A 1.0523, 1.0284, 1.0251, 1.0577, 1.1195, 1.1791, 1.2061, 1.2013,
     B 1.1936, 1.2000, 1.2040, 1.1824, 1.1489, 1.1400, 1.1539, 1.1629,
     C 1.1617, 1.1586, 1.1564, 1.1572, 1.1565, 1.1399, 1.1037, 1.0627,
     D 1.0341, 1.0223, 1.0199, 1.0188, 1.0174, 1.0163  /
      END
```