MODIS
SCIENCE DATA SUPPORT TEAM
PRESENTATION

May 1, 1992

AGENDA

ACTION ITEMS

04/24/92 [Lloyd Carpenter] Prepare the Team Leader's Software and Data Management Plan for review. STATUS: Open. Due Date: May 10, 1992.

04/24/92 [Lloyd Carpenter] Prepare the Team Leader's Science Computing Facility Plan for review. STATUS: Open. Due Date: May 10, 1992.

04/24/92 [Tom Goff] Develop a detailed schedule through to the delivery of Version 1 to the DAAC for Level-1A and -1B software design and development, identification of risk areas in Level-1A and -1B design, and prototyping of risks. STATUS: Open. Due Date:

04/24/92 [J.J. Pan] Develop a detailed schedule for the Level-2 Processing Shell design and development, identification of risk areas in the Level-2 Processing Shell design and development, and prototyping of risks, through to the delivery of Version 1 to the DAAC. Develop a detailed schedule for a typical algorithm integration into the Level-2 processing shell. STATUS: Open. Due Date:

04/24/92 [Lloyd Carpenter & Team] Develop a staffing plan for the accomplishment of the tasks shown on the schedule. STATUS: Open. Due Date:

MODIS Airborne Simulator status (Liam Gumley)

Progress up to 30 April 1992

(1) MAS infrared channel calibration

In order to justify the selection of a blackbody count smoothing/averaging approach, I computed estimates of the sensitivity of the MAS channel 12 detector to changes in the controlled and ambient black body temperatures.

A typical earth scene temperature range is 190 K to 320 K. The MAS sensitivity is adjusted so that radiances within this range span the full range of digitization levels (e.g. 256 or 1024 levels). The corresponding Planck radiances for MAS channel 12 are

$R(190\ K) = 12.40306\ mW\ m^{-2}\ sr^{-1}\ cm^{-1}$,
$R(320\ K) = 165.72116\ mW\ m^{-2}\ sr^{-1}\ cm^{-1}$,

$dR = 165.72116 - 12.40306 = 153.31810\ mW\ m^{-2}\ sr^{-1}\ cm^{-1}$.

Then @ 1024 levels

$dR/dC = 153.31810 / 1024 = 0.14972\ mW\ m^{-2}\ sr^{-1}\ cm^{-1}$ per count.

The temperature controlled blackbody is maintained at approximately 273.15 K, so in this region the Planck radiances at 272 K and 274 K are

$R(272\ K) = 84.37053\ mW\ m^{-2}\ sr^{-1}\ cm^{-1}$,
$R(274\ K) = 87.17340\ mW\ m^{-2}\ sr^{-1}\ cm^{-1}$,

$dR' = 87.17340 - 84.37053 = 2.80287$,
$dT = 274 - 272 = 2$.

Therefore the number of counts over this temperature range (272 K to 274 K) is

$NC = dR' / (dR/dC) = 2.80287 / 0.14972 = 18.72075$,

and the temperature sensitivity is

$TS = dT / NC = 2 / 18.72075 =$ <u>0.10683 K per count</u>.

Similarly, the ambient blackbody stabilizes at approximately 245 K at cruising altitude, so in this region the Planck radiances at 244 K and 246 K are

$R(244\ K) = 50.55198\ mW\ m^{-2}\ sr^{-1}\ cm^{-1}$,
$R(246\ K) = 52.63408\ mW\ m^{-2}\ sr^{-1}\ cm^{-1}$,

$dR' = 52.63408 - 50.55198 = 2.08210$,
$dT = 246 - 244 = 2$.

Therefore the number of counts over this temperature range (244 K to 246 K) is

$NC = dR' / (dR/dC) = 2.08210 / 0.14972 = 13.90663$,

and the temperature sensitivity is

TS = dT / NC = 2 / 13.90663 = 0.14382 K per count.

Shown overleaf are plots of the MAS blackbody temperatures and radiance counts for the MAS flight on 21-NOV-91 (first hour at cruising altitude).

For the ambient black body, the temperature can be seen to decrease from around 246.2 K to 244.1 K over the space of an hour. Given the temperature sensitivity in this temperature region (0.14382 K per count) the counts recorded should have shown a variation of

(246.2 - 244.1 K) / 0.14382 K per count = 14.60159 counts.

Therefore in the absence of noise, the blackbody counts recorded should have shown only about 14 distinct recorded levels. The blackbody count data plotted shows considerably more variation than this. The overall variation during this time is approximately 30 counts, while line to line variations of around $\pm$ 10 counts are observed. The increase in count overall variation is believed to be due to changing instrument sensitivity with time.

This demonstrates that 'smoothing' of the recorded blackbody count data may not be appropriate, since the 'smoothed' data would always contain some remnant of the noise signal. A better approach would be to fit a simple straight line or low-order curve to the black body data over a given time interval. A flight track is the obvious time period to use for this fit. Therefore it is recommended that either a straight line, or 2nd degree polynomial be fitted to the blackbody count data for each flight track. This will adequately represent the real variation in the blackbody temperatures, and remove the effects of the single sample detector noise in the blackbody counts.

(2) MAS data usage by Dorothy Hall's group

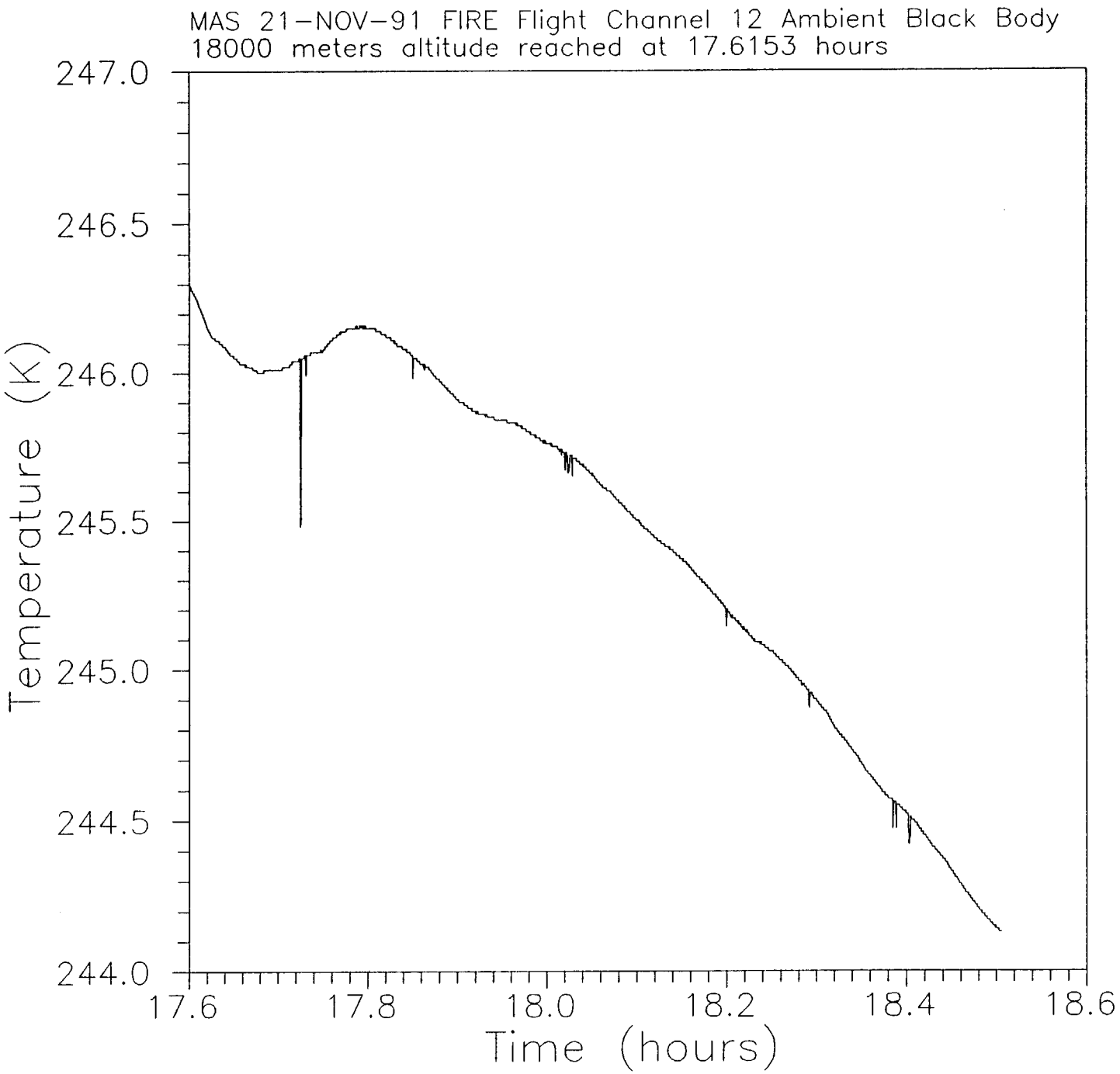
Last Friday (04/24) I spent an hour with Dorothy and one of her programmers to test whether EASI/PACE was reading the MAS data correctly. I was able to verify that the MAS subset utility was creating flat image files correctly, however EASI/PACE was having difficulty reading the flat images correctly. Images read were displayed upside down, and horizontal striping was added to portions of the image. Not being an EASI/PACE expert, I suggested that they call PCI to try and solve this problem.

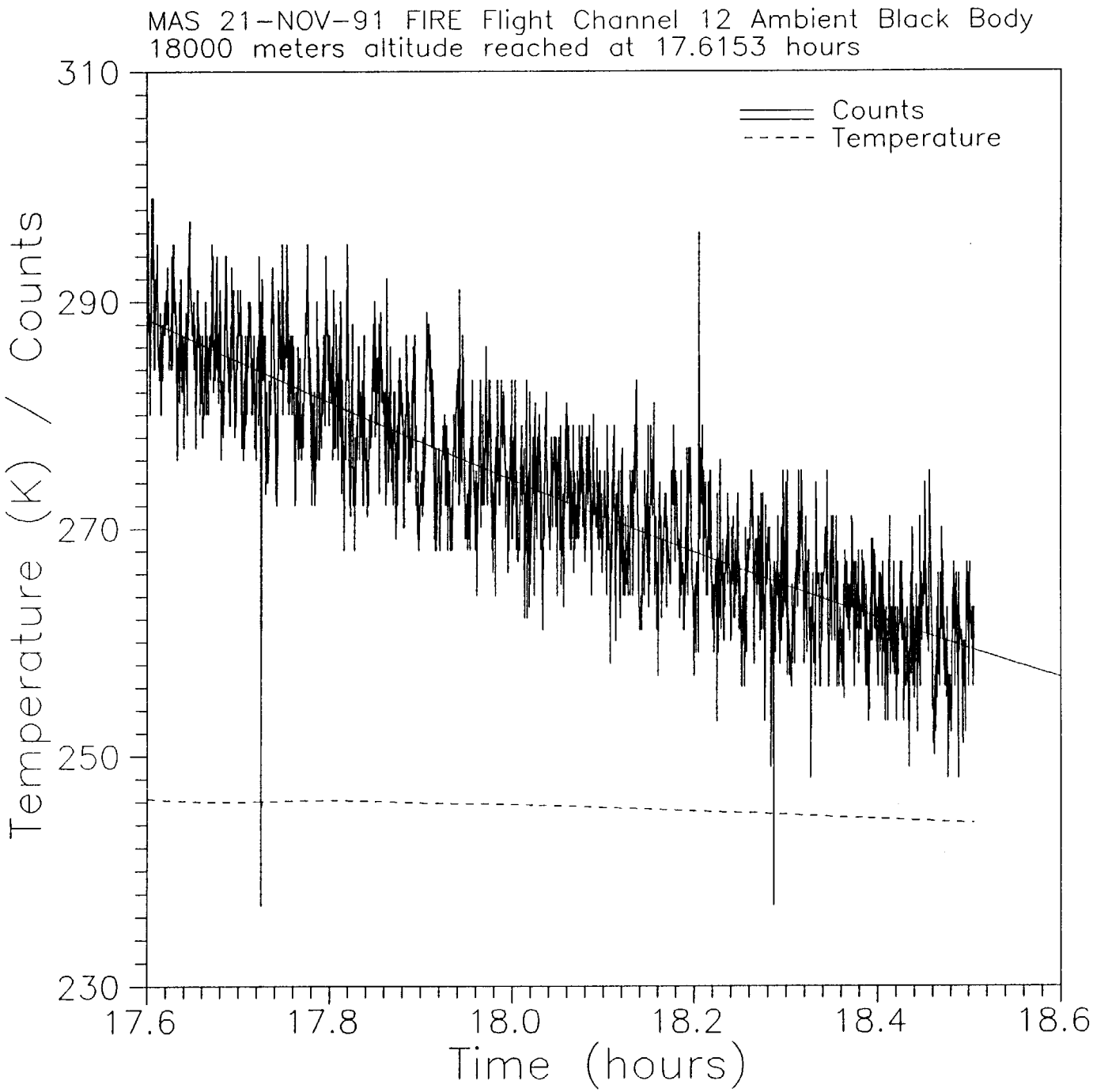
(3) Demonstration of MAS processing to EOSDIS personnel

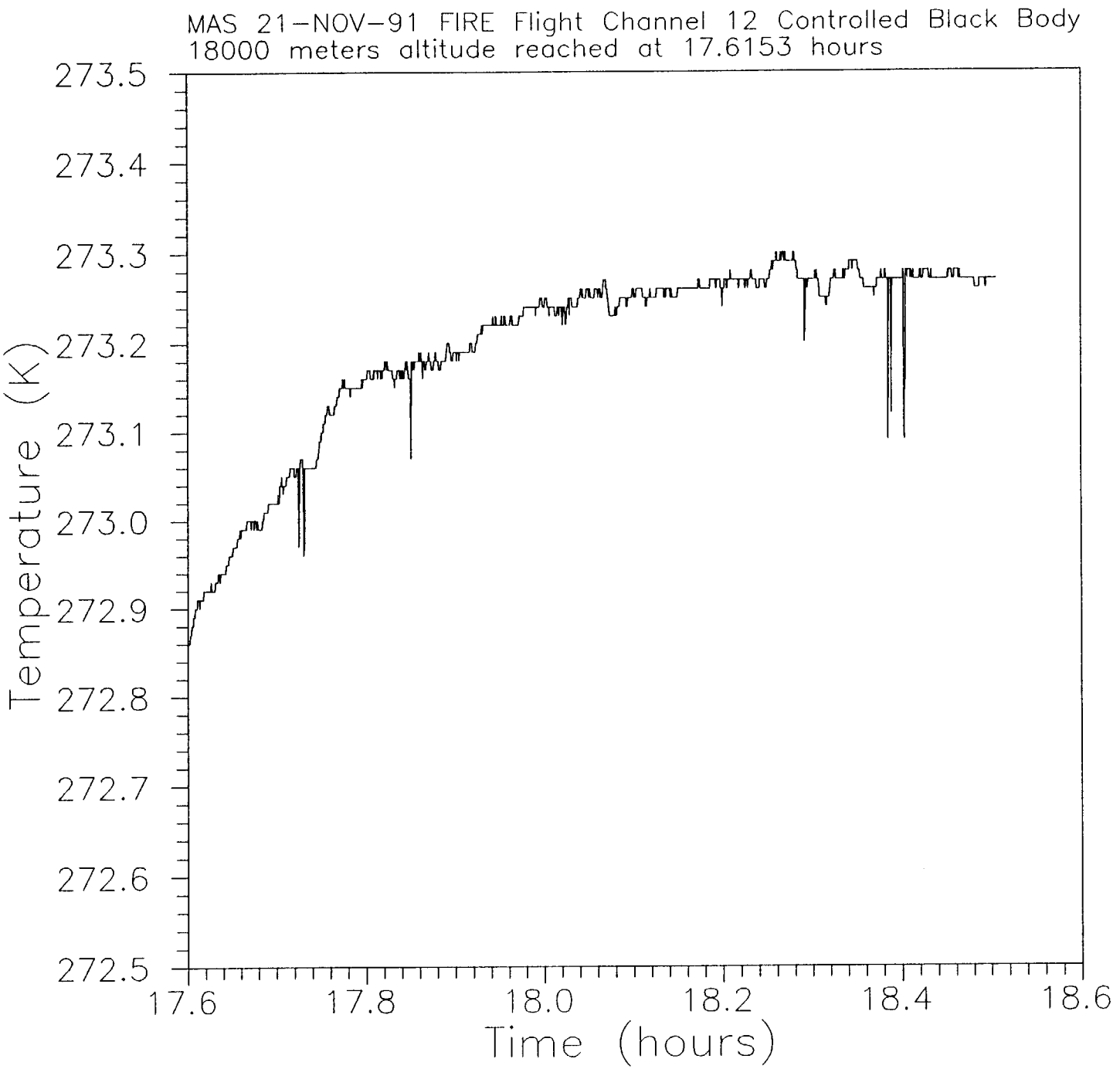
On Wednesday (04/29) I spent some time demonstrating the MAS Level-1 processing system to Pershing Anderson, who works for Sol Broder. He did not have much information about the MAS, so I provided him with the following reports:
MAS Level-1 Processing System Status and Data Availability (04/13/92),
Investigation of a cataloguing scheme for the MODIS Airborne Simulator (MAS) (02/20/92),
Investigation of strategies for storing and distributing MAS data (02/20/92).
He has not seen the demonstration software package yet, but will be in touch with me when he obtains a copy from Sol.

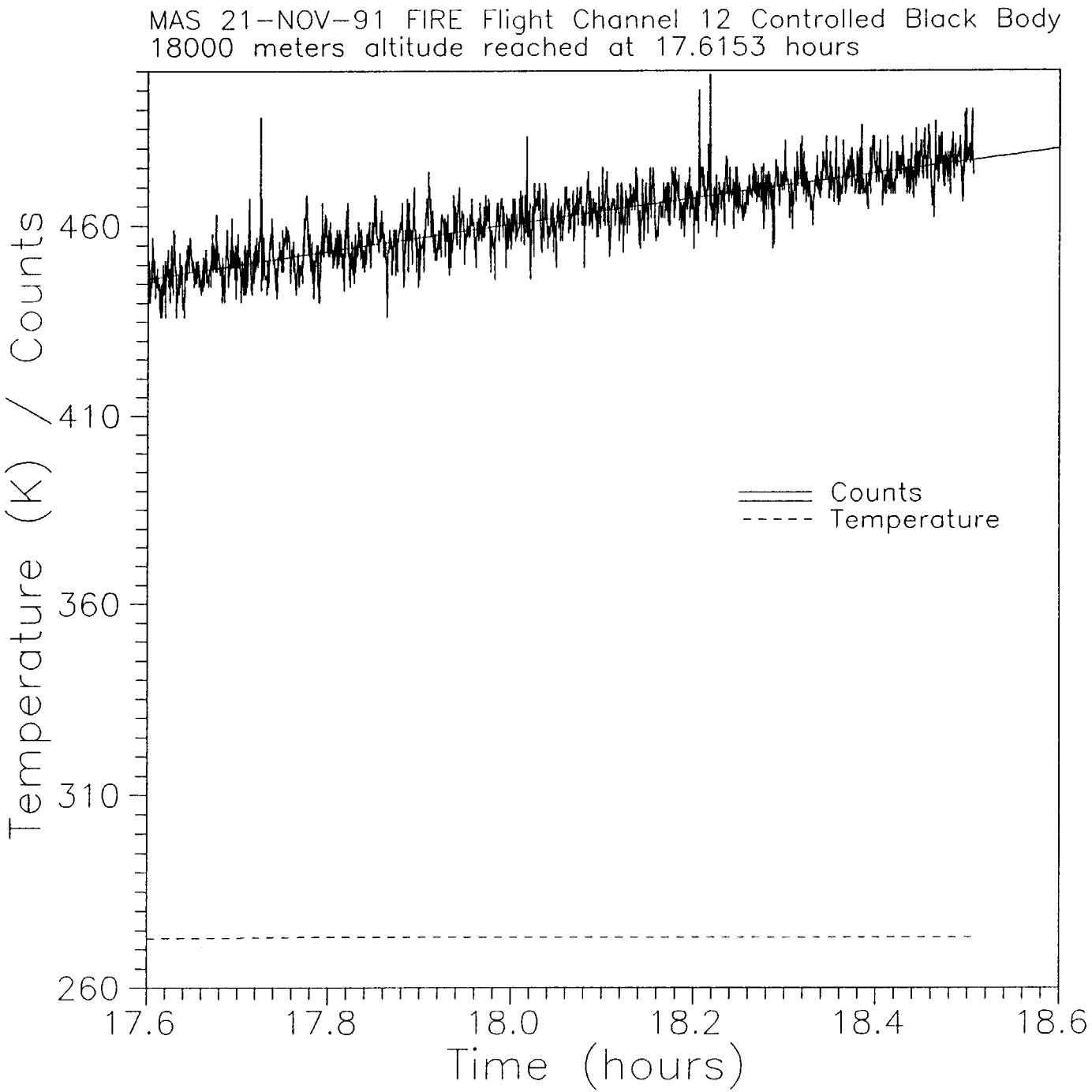
(4) MAS integration and testing at Ames

After speaking to Chris Moeller, my tentative plan is to be at Ames from 05/21/92 to 05/23/92. A MAS test flight is currently planned for 05/21/92.


MAS 21-NOV-91 FIRE Flight Channel 12 Ambient Black Body
18000 meters altitude reached at 17.6153 hours
247.0
246.5
246.0
245.5
245.0
244.5
244.0
Temperature (K)
17.6
17.8
18.0
18.2
18.4
18.6
Time (hours)


MAS 21-NOV-91 FIRE Flight Channel 12 Ambient Black Body
18000 meters altitude reached at 17.6153 hours
Counts
Temperature
Temperature (K) / Counts
310
290
270
250
230
17.6
17.8
18.0
18.2
18.4
18.6
Time (hours)


MAS 21-NOV-91 FIRE Flight Channel 12 Controlled Black Body
18000 meters altitude reached at 17.6153 hours
Temperature (K)
273.5
273.4
273.3
273.2
273.1
273.0
272.9
272.8
272.7
272.6
272.5
17.6
17.8
18.0
18.2
18.4
18.6
Time (hours)


MAS 21-NOV-91 FIRE Flight Channel 12 Controlled Black Body
18000 meters altitude reached at 17.6153 hours
Temperature (K) / Counts
460
410
360
310
260
17.6
17.8
18.0
18.2
18.4
18.6
Time (hours)
Counts
Temperature

MODIS Science Data Support Team (SDST)
Coding Recommendations for the MODIS Science Team
(DRAFT)

J. J. Pan
Research and Data Systems Corporation
(301) 982-3738

A Fortran program, called RADIANCE, is being prepared for the MODIS Science Team Members. The flow chart of this program and part of the source code (three subroutines) are attached.

RADIANCE, originally coded by Liam Gumley, can be used to compute the sensor weighted radiance of a black body source. There are one main program and five major subroutines:

RADIANCE	: Main program; input the spectral band number,
RADCALC	: Subroutine used to initialize variables and link other subroutines,
OPENA	: Subroutine used to open data files,
PLANCK	: Subroutine used to compute the Planck radiance,
SPLINE	: Subroutine used to compute cubic spline interpolation,
INTEGRAL	: Subroutine used to perform numerical integration.

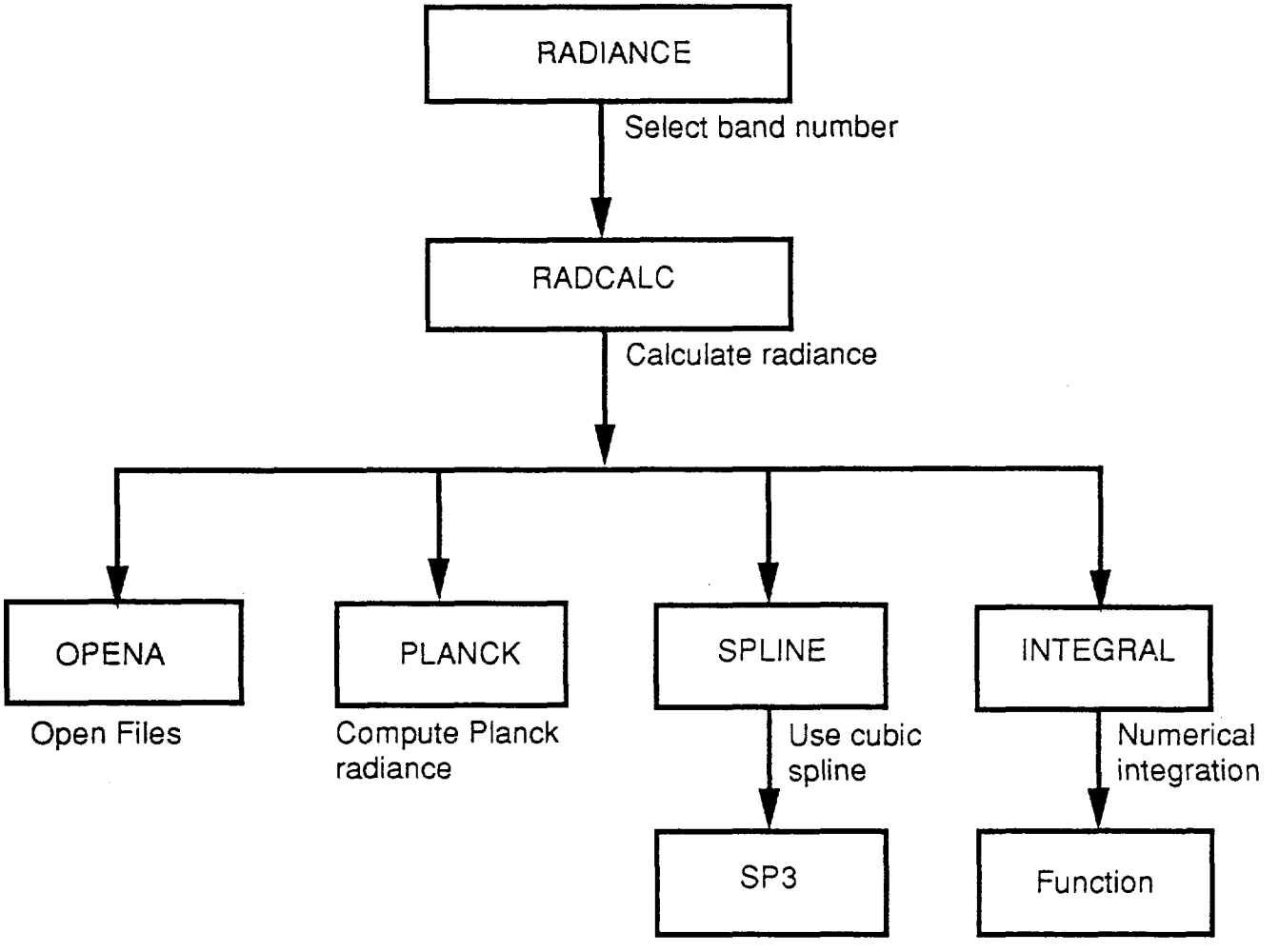
Currently, each subroutine is compiled using Microsoft Fortran Compiler Ver. 5.1 on PC, then FTNCHEK, a Fortran program checker, is used to detect semantic errors. Two options, -f77 and -port, of FTNCHEK, are selected in the test,

-f77	Warn about violations of the Fortran 77 standard,
-port	Warn about non-portable usages.

The command and test results from FTNCHEK are also attached.

It is expected to finish the source coding in one week, which will include the syntax and semantic errors detection. The next task is to refine the structure, highlight some potential problems, and improve its readability. An internal review of this program will be scheduled after the preliminary test.

Flow Chart of Program RADIANCE



```
c-----------------------------------------------------------------------
c
c  Program:     RADIANCE
c
c  Purpose:     To compute the sensor weighted radiance of a black
c               body source.
c
c  Reference:   Planet, W. G., 1988, Data extraction and calibration of
c               TIROS-N/NOAA Radiometers. NOAA Tech. Memo. NESS 107 -
c               Rev. 1, Washington DC.
c
c  Version:     2.0 (4/30/92)
c
c  Language:    Fortran 77
c
c  System:      PC using Microsoft FORTRAN Ver.5.1
c
c  Revisions:    REV.   DATE       BY            REASON
c               ------ ---------  ----------   ------------------------------
c                1.0    8-27-91   L. Gumley
c                2.0    4-30-92   J. Pan, L. Gumley,
c                                 L. Carpenter   Code recommendation for MODIS
c
c  Subroutines Called:  RADCALC
c
c  Common Block:        None.
c
c  External Function:   None.
c
c  Declarations:
c   -- Input --
c      integer   band  -- The band number
c
c-----------------------------------------------------------------------
c  Sponsor:     National Aeronautical and Space Administration (NASA)
c               Goddard Space Flight Center (GSFC)
c               Moderate Resolution Imaging Spectrometer (MODIS)
c               Science Data Support (SDST)
c               Dr. William Webster (301) 286-4506
c               Code 920.2, Greenbelt, MD, U.S.A. 20771
c
c  Author:      L. Gumley, J. J. Pan, L. Carpenter
c               Research and Data Systems Corporation (RDC)
c               7855 Walker Drive, Suite 460, Greenbelt, MD 20770-3212
c               (301) 982-3700
c
c  Notice:  This software may be freely used and distributed without any
c           compensation to the author or sponsor. It is provided without
c           support and without any obligation, whatever, to assist in its
c           use, correcting, and modification, or enhancement.
c
c  THIS SOFTWARE IS PROVIDED "AS IS" WITH NO EXPRESS OR IMPLIED WARRANTIES
c  OF ANY KIND, INCLUDING WARRANTIES OF DESIGN, MECCHANTIBILITY OR FITNESS
C  FOR A PARTICULAR PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE, OR
C  TRADE PRACTICE. There is no warrenty that this software will meet any
c  particular  specification nor is there any warrenty that the documentation
c  providing instructions or information for use of the software is accurate
c  or otherwise conforms to the software requirements. Further, in furnishing
c  this software, there shall be no liability, under any circumstances, for
c  either direct or consequential damages.
c
c  Users of this software agrre that they will retain this Notice, in its
c  entirety, within the software source listing and documentation, prior
c  to the distribution of this software to other parties.
```

```
c
c-------------------------------------------------------------------------------

      integer band

      write(*,*) ' Enter the band number :'
      read(*,*, err=99)  band

      call radcalc(band)
      stop

99    write(*,*) ' Error : Invalid band number.....'

      end
```

Use FTNCHEK to check a Fortran program:

```
-----------------------------------------------------------------------
  Command: ftnchek -f77 -port radiance.for
           ( -f77:  warn about violations of the Fortran 77 standard,
             -port: warn about non-portable usages.)
-----------------------------------------------------------------------

Output:

FTNCHEK Version 2.5 January 1992

File radiance.for:

Names longer than 6 chars in module %MAIN (nonstandard):
   RADCALC


 0 syntax errors detected in file radiance.for
 1 warning issued in file radiance.for

Subprogram RADCALC never defined
    invoked in module %MAIN line 73 file radiance.for
```

```
c-----------------------------------------------------------------------
c   Subroutine: RADCALC
c
c   Version:    2.0 (4/30/92)
c
c   Language:   Fortran 77
c
c   Purpose:    For a given spectral band, read temperature data from a
c               file, compute the corresponding Planck radiance, and store
c               the output to a file.
c
c               Temperature range is 150K to 373K in 1K steps.
c
c   Reference:  Planet, W. G., 1988, Data extraction and calibration of
c               TIROS-N/NOAA Radiometers. NOAA Tech. Memo. NESS 107 -
c               Rev. 1, Washington DC.
c
c   Revisions:   Rev.    Date       By          Reason
c               -----  --------- --------   --------------------------
c                1.0    7-14-91   L. Gumley
c                2.0    4-30-92   J. Pan, L. Gumley,
c                                  L. Carpenter  For MODIS Code Recomm.
c
c   Subroutine Called:    Opena, Planck, Spline, Integral
c
c   Common Block:         Splcom
c
c   External Function:    Funca, Funcb
c
c   Declarations:
c    -- Data File --
c       SRF-??.ASC        -- An ASCII data file where ?? = band.  This should
c                            contain two columns of real numbers. The first
c                            should be wavenumber in inverse centimeters.
c                            The second should be spectral response, which
c                            has no units, and may or may not be normalized.
c
c       TTR-??.ASC        -- An ASCII data file where ?? = band. This will
c                            contain two columns of real numbers. The first
c                            will be temperature in Kelvin. The second will be
c                            the corresponding Planck radiance weighted
c                            by the spectral response in milliWatts per square
c                            meter per steradian per wavenumber.
c
c    -- Input --
c       integer  band     -- Spectral band for which to compute Planck
c                            radiances.
c
c    -- Output --
c       real     answer -- Numerical integral value
c       integer  ifail  -- Error flag (0 = success, 1 = fail)
c       integer  number -- Number of intervals in final
c                          iteration of the trapezoid rule
c
c-----------------------------------------------------------------------
      subroutine radcalc(band)

      integer          band

c-----internal variables
      parameter        (nsize = 500)
      character        filename*72, action*3
```

```
c-----arrays used for spline functions
      real             x(nsize), y(nsize), h(nsize), b(nsize),
     &                 u(nsize), v(nsize), z(nsize)

c-----temperature and count variables
      real             temp
      integer          itemp, itemp1, itemp2, itemps
      integer          icount
c-----parameters for file open/close
      integer          device, ifail

      common           /splcom/ icount, x, y, z, temp
      real             funca, funcb
      external         funca, funcb


c-----open spectral response file
      write( filename, '( ''srf-'', i2.2, ''.asc'' )' ) band

      device = 21
      action = 'old'
      call opena( device, filename, action, ifail )

      if (ifail .ne. 0) then
          write(*,*) ' Error opening SRF-??.ASC file in RADCALC ..'
          stop
      endif

c-----read x-y data from file
      icount = 0

      do 60 i = 1, 100
         read( device, *, end = 80 ) x( i ), y( i )
         icount = icount + 1
60    continue

80    close(device)

c-----compute spline coefficients for spectral response
      call spline(icount, x, y, h, b, u, v, z)

c-----define temperature range
      itemp1 = 150
      itemp2 = 373
      itemps = 1

c-----compute integral of spectral response function
      call integral( funca, x(1), x(icount), 1.0e-05, a1, n1, ifail)

      if (ifail .ne. 0) then
          write(*,*) ' Error computing integral in RADCALC..'
          stop
      endif

c-----open output TTR-??.ASC file
      write( filename, '( ''ttr-'', i2.2, ''.asc'' )' ) band
      action = 'new'
      call opena( device, filename, action, ifail )

      if (ifail .ne. 0) then
          write(*,*) ' Error opening TTR-??.ASC file in RADCALC..'
          stop
      endif
```

```
      do 140 itemp = itemp1, itemp2, itemps
         temp = float(itemp)
c--------compute integral of spectral response times Planck function
         call integral( funcb, x(1), x(icount), 1.0e-05, a2, n2, ifail)

         if (ifail .ne. 0) then
            write(*,*) ' Error computing integral in RADCALC..'
            stop
         endif

         rad1 = a2 / a1
         write( device, '( 1x, 4( f10.5, 2x ), i6 )' ) temp, rad1
140   continue

      close(device)

      return
      end
```

Use FTNCHEK to check a Fortran program:

```
-----------------------------------------------------------------------
  Command: ftnchek -f77 -port radcalc.for
           ( -f77:  warn about violations of the Fortran 77 standard,
             -port: warn about non-portable usages.)
-----------------------------------------------------------------------

Output:

FTNCHEK Version 2.5 January 1992

File radcalc.for:

Names longer than 6 chars in module RADCALC (nonstandard):
  FILENAME            INTEGRAL             RADCALC


 0 syntax errors detected in file radcalc.for
 1 warning issued in file radcalc.for

Subprogram RADCALC never invoked
    defined in module RADCALC line 56 file radcalc.for
Subprogram OPENA never defined
    invoked in module RADCALC line 120 file radcalc.for
Subprogram SPLINE never defined
    invoked in module RADCALC line 102 file radcalc.for
Subprogram INTEGRAL never defined
    invoked in module RADCALC line 130 file radcalc.for
Subprogram SP3 never defined
    invoked in module FUNCA line 156 file radcalc.for
    (possibly it is an array which was not declared)
Subprogram PLANCK never defined
    invoked in module FUNCB line 171 file radcalc.for
    (possibly it is an array which was not declared)
```

```
c-----------------------------------------------------------------------
c   Subroutine: OPENA
c
c   Version:    2.0 (4/30/92)
c
c   Language:   Fortran 77
c
c   Purpose:    Open a new or old ASCII file for sequential
c               access, with pointer positioned at start.
c
c   Reference:  Microsoft Fortran Ver. 5.1 Reference
c
c   Revisions:   Rev.    Date       By          Reason
c               -----  --------- --------    -------------------------
c                1.0    8-26-91   L. Gumley
c                2.0    4-30-92   J. Pan, L. Gumley
c                                  L. Carpenter  For MODIS Code Recomm.
c
c   Subroutine Called:    None.
c
c   Common Block:         None.
c
c   External Function:    None.
c
c   Declarations:
c    -- Input --
c       integer      device    -- Fortran device number
c       character*72 name      -- Name of file to open
c       character*3  action    -- 'NEW' or 'OLD'
c
c    -- Output --
c       integer      ifail     -- 0 = success, 1 = failure
c
c-----------------------------------------------------------------------
      subroutine opena (device, name, action, ifail)

      integer          device, ifail
      character*72     name
      character*3      action

c-----Check file status
      if ( action .eq. 'new' ) then
           open( device, file = name, status = 'unknown',
     &           form = 'formatted', access = 'sequential', err = 20 )
        else
           open( device, file = name, status = 'old',
     &           form = 'formatted', access = 'sequential', err = 20 )
      endif

      rewind(device)
      ifail = 0
      return

20    ifail = 1

      return
      end
```

Use FTNCHEK to check a Fortran program:

```
------------------------------------------------------------------------
  Command: ftnchek -f77 -port opena.for
           ( -f77:  warn about violations of the Fortran 77 standard,
             -port: warn about non-portable usages.)
------------------------------------------------------------------------

Output:

FTNCHEK Version 2.5 January 1992

File opena.for:


 0 syntax errors detected in file opena.for

Subprogram OPENA never invoked
    defined in module OPENA line 35 file opena.for
```

```
c-----------------------------------------------------------------------
c   Function:    PLANCK
c
c   Version:     2.0 (4/30/92)
c
c   Language:    Fortran 77
c
c   Purpose:     To compute the Planck radiance at a given temperature and
c                wavenumber, as per NOAA/NESDIS Tech. Memo. 107.
c
c   Reference:   Planet, W. G., 1988, Data extraction and calibration of
c                TIROS-N/NOAA Radiometers. NOAA Tech. Memo. NESS 107 -
c                Rev. 1, Washington DC.
c
c   Revisions:    Rev.     Date        By           Reason
c                -----  --------- --------    -------------------------
c                 1.0    7-14-91   L. Gumley
c                 2.0    4-30-92   J. Pan, L. Gumley
c                                  L. Carpenter  For MODIS Code Recomm.
c
c   Subroutine Called:     None.
c
c   Common Block:          None.
c
c   External Function:     None.
c
c   Declarations:
c    -- Input --
c       real     temp         -- Emission source temperature (K)
c       real     waveno       -- Emission wavenumber (1/cm)
c
c    -- Output --
c       real     planck       -- Planck radiance (milliWatts per square
c                                meter per steradian per wavenumber)
c
c-----------------------------------------------------------------------
      real  function  planck(temp, waveno)

      real  temp, waveno, c1, c2

      c1 = 1.1910659e-05
      c2 = 1.4388330

      planck = c1 * waveno**3 / (exp(c2 * waveno/temp) - 1.0 )

      return
      end
```

Use FTNCHEK to check a Fortran program:

```
-----------------------------------------------------------------------
  Command: ftnchek -f77 -port planck.for
           ( -f77:  warn about violations of the Fortran 77 standard,
             -port: warn about non-portable usages.)
-----------------------------------------------------------------------

Output:

FTNCHEK Version 2.5 January 1992

File planck.for:


 0 syntax errors detected in file planck.for

Subprogram PLANCK never invoked
    defined in module PLANCK line 37 file planck.for
```

Cadre and NetCDF
Thomas E. Goff
30 April, 1992

teg@cheshire.gsfc.nasa.gov,
(301) 982-3704
tgoff on GSFC mail

* **Cadre's Teamwork** - Ed Masuoka had a discussion with the sales representative from Cadre to clear up several items about our potential use of this software product. Items discussed were as follows:

  - Teamwork runs under X-11 windows rev 4 (X11.4). It will work as an X11.4 client on the Sun Sparc workstation. It requires about 50 MBytes of disc space and 8 MBytes per X11.4 user of system memory. The license floats - only concurrent users are counted, not machines or individuals. A two person core license with single person editor licenses seams judicious at the present time.

  - Teamwork can be transferred from the Sun to the future TLCF HP 730 for the cost of media and manuals ($500).

  - Training is available in Providence, Rhode Island for three days. If we act quickly, there will be a training session on site at GSFC in June courtesy of code 560. Note that Teamwork training is not the same as Object Oriented Programming (OOP) training or C++ training.

  - C++ code generation is supported as an add-on and can be used concurrently with C. FORTRAN code generation is not currently supported. Reverse engineering of C, FORTRAN, and C++ is supported. A separate object oriented module is available and recommended for purchase.

  - Support is available for a small extra cost that allows people to phone Cadre via an 800 number for help during normal working hours. This is indispensable and highly recommended. Upgrade maintenance is included in this service.

  - Teamwork will format and print a data dictionary, but they recommend a publishing package to integrate diagrams and text to produce a finished document. Framemaker appears to be the best choice for this desk top publishing (dtp) function.

  - Teamwork supports configuration management (CM) of the structure chart and dictionary elements, but a third party CM tool is recommended for code management.

  - Teamwork supports the normal vendor supplied compilers as well as Centerline's (ex Saber) C and C++ environment.

* **NetCDF installation on the Mac** - NetCDF uses the underlying eXternal Data Representation (XDR) library as the machine independent method for interchanging binary data (integer, floating point, and characters) among differing machine architectures. This is the library that is used by TCP/IP, rpc, telnet, etc. The sources were placed in the public domain in 1984 by Sun Computers. The NetCDF distribution uses the XDR library that is resident on (almost) all unix machines. The Apple Mac machines (to the best of my knowledge) do not have a resident XDR library. This must be built from the source files included in the NetCDF distribution. I have attempted to compile these files including the xdrtest routine written by the University Corporation for Atmospheric Research (ucar) on a Sun computer, an IBM-PC clone, and the Apple Macintosh. The files, as supplied, will compile on the Sun, but not on the PC or Mac. A syntax error in xdr_stdio was easily fixed, but a conflict between and array address and a pointer type in a second order macro in xdrtest was harder to find. These modified routines will now compile on the Sun and Mac computers.

  Supplied with the XDR routines is a binary XDR stream file with properly encoded data to be used as a comparison with an output stream from the newly built XDR routines. If the XDR routines are built on the Sun using the supplied Makefile, the binaries compare. If the XDR routines are built by hand on the Sun, the binary files do not compare. This problem needs to be understood and exorcised. The PC versions also do not compare. The Mac version has yet to be performed.

  I have been in contact with a contractor to the USGS at Woods Hole, Chuck Denham, who has successfully generated the C version of the NetCDF library. This was performed on a Mac-II using the Macintosh Programmers Workbench (MPW) with C compiler. This MPW environment has now been installed on a Mac in Mike King's shop. Chuck has generously supplied a two page write up with helpful hints about what he did to perform the port in addition to machine readable MPW make files. I will attempt the installation as soon as possible to allow Mike King's Macintoshes to be bench marked for his future processing requirements.

* **TLCF Considerations** - Maximum productivity for generating the MODIS production programs can best be achieved by keeping a few consideration in mind:

  - If a networked environment is to be used in which X-11 windows with Motif (or similar) will be the major graphics user interface (GUI - pronounced gooey), a network filter bridge between the local networked machines and the remaining of the GSFC network will greatly increase the response time for the local users without compromising network capabilities. A filter bridge will block outside network traffic from reaching the local network as well as preventing local traffic from competing with the GSFC network bandwidth. Also, thin LAN is configurable by the users, while thick LAN (EtherNet) must be installed and attended by the GSFC network branch.

- A local post script printer, accessible without leaving your chair is a low cost convenience. For example: an HP LaserJet IIIP with the new HP postscript-II cartridge can be purchased for less than $2000, including a memory upgrade.

- It is desirable to have direct phone lines to the host to allow character based modem and slip connections to be available. These should be directly connected to the Rolm phones.

- The computer must have an ANSI C compiler, C++ 2.0 compiler, and FORTRAN90 when available. This seems like a trivial requirement given the fact that ANSI C has been available for quite some time, but Sun computers do not support ANSI C. I requested, and Karl Anderson is currently installing, the gnu C and C++ compilers (GCC) on the cheshire Sun computer. The fdump and companion routines, which are ANSI compliant, will be tested using GCC on the Sun in addition to the SGI, PCs, and Mac.

- MOST IMPORTANT: The TLCF should have a full maintenance contract with the vendor that allows direct contact with the vendor via an 800 number and a guaranteed minimum response time in minutes. This allows users (the people who program and use the machine on a continuing daily basis) to resolve problems quickly before they become schedule slippers. Included in this maintenance are operating system and manual updates.

-- Miscellaneous --

& **Phone/SLIP Connection** - The SGI cdc910b16 iris now has a direct phone connection that works in character mode. This connection will be used as a slip connection when time allows me to obtain the iris slip manual and configure my PC-NFS appropriately. When this is successful, an X-11 rev 4 windows server for PC's would be a nice capability for all users.

& **FORTRAN Source Code Checking** - FORTRAN LINT (flint) is currently on the ltp iris and needs to be exercised to compare this program with PR:QA and ftncheck. Our evaluation copy has been extended another two weeks. Perhaps someone in code 560 is interested in evaluating this program?

& **Anchor Point Prototyping** - Deriving the location of MODIS pixels on the Earth surface is a desired component of the MODIS level-1b processing software. An early prototyping effort towards that end could be initiated that will result in a series of plots, in different projections, of the proposed MODIS IFOV. This would be of great use to the Team Members and Science Data Support Team in understanding the overlap and pixel mixing problems of the MODIS pixels due to the "bow tie" effect at the off-nadir scan positions.