Federal Aviation Administration

DOT/FAA/AM-14/13
Office of Aerospace Medicine
Washington, DC 20591

CARI-NAIRAS: Calculating Flight Doses From NAIRAS Data Using CARI

Kyle Copeland
Civil Aerospace Medical Institute
Federal Aviation Administration
Oklahoma City, Oklahoma 73125

Christopher Mertens
Langley Research Center
National Aeronautics and Space Administration
Hampton, VA 23681

December 2014

Final Report

NOTICE

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/AM-14/13		

4. Title and Subtitle	5. Report Date
CARI-NAIRAS: Calculating Flight Doses from NAIRAS Data using CARI	December 2014
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Copeland K,[1] Mertens C[2]	

9. Performing Organization Name and Address		10. Work Unit No. (TRAIS)
[1]FAA Civil Aerospace Medical Institute P.O. Box 25082 Oklahoma City, OK 73125	[2]National Aeronautics and Space Administration Langley Research Center Hampton, VA 23681	11. Contract or Grant No.

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Office of Aerospace Medicine Federal Aviation Administration 800 Independence Ave., S.W. Washington, DC 20591	14. Sponsoring Agency Code

15. Supplemental Notes

Work was accomplished under approved tasks AM- TOXLAB.AV9000 and PSRLAB.AV9100

16. Abstract

The CARI computer program is galactic cosmic radiation (GCR) dose calculation software developed by the U.S. Federal Aviation Administration. It serves the aerospace industry and flying public by providing a means of calculating GCR doses for flights, and as such, is a valuable radiation monitoring tool aiding industry and individuals in their radiation protection efforts. The information the software provides is also used by research scientists to investigate health effects of chronic exposure to low levels of ionizing radiation present in the atmosphere. CARI-6 and previous versions were increasingly inaccurate above 60,000 feet because of the superposition approximation built in to their global dose rate tables.

This report describes CARI-NAIRAS, a new version of CARI that uses pre-calculated global tables of dose rates generated by the NAIRAS (Now-Cast of Atmospheric Ionizing Radiation for Aviation Safety) system developed at National Aeronautics and Space Administration (NASA) Langley Research Center. The NAIRAS system uses the NASA radiation transport code HZETRN (High charge (Z) and Energy TRaNsport), which does not use the superposition approximation, as well as satellite and ground-based data inputs to generate the global tables. CARI-NAIRAS is shown to be in good agreement with Monte Carlo based calculations in the altitude range 27,000 to 87,000 feet, thus eliminating the need for the altitude limit of 60,000 ft. Flight dose estimates are similar to those of CARI-6 and CARI-7. For 24 of the 32 flights investigated, CARI-NAIRAS estimated an effective dose within 20% of the mean of the three programs (CARI-6W, CARI-7, and CARI-NAIRAS). CARI-NAIRAS estimates are expected to improve once the latest version of HZETRN is incorporated into NAIRAS.

17. Key Words	18. Distribution Statement
CARI, Galactic Cosmic Radiation, HZETRN, In-Flight Radiation, Ionizing Radiation, NAIRAS, Space Weather	Document is available to the public through the Internet: www.faa.gov/go/oamtechreports

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	64	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Contents

CARI-NAIRAS: Calculating Flight Doses
From NAIRAS Data Using CARI

CARI-NAIRAS: Calculating Flight Doses From NAIRAS Data Using CARI

I. BACKGROUND

In-flight exposure to ionizing radiation has been a health concern for passengers and crewmembers since the early days of jet travel. The U.S. Federal Aviation Administration (FAA) established the Radiobiology Research Team at its newly founded Civil Aeromedical Research Institute (now called the Civil Aerospace Medical Institute, i.e., CAMI) to investigate the health effects of ionizing radiation in 1961.

By the late 1980s, there was an obvious need for user-friendly software that would calculate a reasonably accurate flight dose and run on a personal computer (PC). It would be valuable as a tool to scientists investigating the health effects of radiation exposure and as an aid to medical officers monitoring radiation exposures of flight-crew members. If it was to be used for estimating career doses of multiple pilots, the software would need to be much faster than the radiation transport codes of the day. The available radiation transport codes were too time consuming to use directly. For example, with LUIN, [1] one of the fastest atmospheric ionizing radiation codes of the time, to calculate the dose rate at a single point in time and space on an IBM personal computer with an Intel 80286 processor required over 12 minutes of calculations. Calculating a dose for even a short flight route would take hours. The development of the software now called CARI shortened that time to a few seconds.

The CARI computer software suite was developed at CAMI and estimates galactic cosmic radiation received by travelers during air travel. The program has had four major revisions since 1991. Originally, the programming was in Microsoft's QuickBASIC and the program was called "Carrier." The name was later shortened to "CARI," echoing the old acronym for the Civil Aerospace Medical Institute. Carrier was based upon the work of Schaefer.[2] CARI-2 through CARI-6 used databases of LUIN dose rate calculations, with each major revision following a major revision of LUIN. Table 1 lists the major releases of CARI, along with the source used for radiation transport and the year released.

The major release versions of CARI were often altered to suit the needs of specific users (e.g., the U.S. Air Force and the U.S. National Institute of Occupational Safety and Health). Thus, there were two versions of CARI-2 (2 and 2N), four versions of CARI-3 (3, 3C, 3N, and 3Q), three versions of CARI-4 (4/4EZ, 4Q, 4R), four versions of CARI-5 (5, 5E, 5AF, and 5G), and seven versions of CARI-6 (6, 6HF, 6M, 6P, 6PM, -6W and -WEB).

With each major revision of LUIN, a version of CARI (designated by the -L, for LUIN) was developed incorporating the associated version of LUIN directly into the code. This approach eliminated the need for databases--but also resulted in a much longer run time. The CARI-L programs were used to check the associated database-using programs and included CARI-L94 (CARI-3 series), CARI-L97 (CARI-4 series), CARI-LF (CARI-5 series), and CARI-LF2, -LF3, and -LF4 (CARI-6 series). Because of distribution limitations placed on LUIN by Professor O'Brien, these versions were and are not distributed without his express permission.[9]

The most significant change in CARI output was the conversion to effective dose calculations in CARI-5. CARI-4 calculated equivalent dose to bone marrow and skeletal tissue based on data from a 30-cm tissue equivalent slab phantom. CARI-5 calculated effective dose following the recommendations in International Commission on Radiological Protection (ICRP) Publication 60[10] (and modified slightly in subsequent publications) by incorporating the fluence-to-effective dose conversion factors collected by Pelliccioni.[11] CARI-6, CARI-7,[12] and CARI-NAIRAS incorporate these conversion factors. CARI-6W,[13] CARI-7, and CARI-NAIRAS also calculate other endpoints such as ambient dose equivalent H*(10).[14]

The major limitation of CARI-6 is accuracy above 60,000 ft. The most common particles in the primary cosmic radiation (referred to as "primaries" because they have not yet undergone a collision resulting in nuclear break-up) are protons, but there are significant numbers of alpha particles and heavier ions, referred

Table 1. Radiation transport sources used by CARI major releases

Program	Radiation Transport	Year Released
CARRIER/CARI	Scheafer, 1971	1989
CARI-2	LUIN[3]	1991
CARI-3	LUIN94[4]	1995
CARI-4	LUIN97	1997
CARI-5	LUIN98f	1998
CARI-6	LUIN99/LUIN2000[5]	2000
CARI-7	MCNPX 2.7.0[6]	2014
CARI-NAIRAS	HZETRN[7,8]	2014

to collectively as "heavy ions" or "HZE" particles (high Z and E, where Z and E are commonly used in nuclear physics as symbols for nuclear charge and energy). The interactions of these particles with the atoms in the atmosphere are extremely difficult to model. Above about 60,000 feet, as altitude increases, accurate modelling of HZE particle interactions becomes increasingly important. This is true, particularly with respect to effective dose, because of the way it is calculated.[10,15,16] LUIN avoided the difficulty of directly modelling HZE particle interactions by employing what is often referred to as the "superposition approximation." This approximation converts the HZE particles into collections of protons and neutrons with the same energy per nucleon as the HZE primary particle. While the approximation can be used to make an accurate picture of the particle mix in the atmosphere at altitudes below 60,000 ft., the number of HZE particles in the atmosphere increases as altitude increases. For altitudes where HZE particles are important, a different approach was needed. Because NAIRAS[17] (see Appendix A for a brief description) is essentially unlimited in altitude (up to 90 km, equivalent to about 300,000 ft.), CARI-NAIRAS was developed as one of two alternatives to enable more accurate dose assessments at the high altitudes needed for commercial space flights (0 to 300,000 ft.) and for near-space, high-altitude, manned balloon flights (80,000 to 120,000 ft.) already being offered to the public.

II. DEVELOPMENT OF CARI-NAIRAS

CARI-NAIRAS is similar to CARI-6W in most respects, except that the dose rates are taken from NAIRAS annual-average Tables calculated with HZETRN,[7,8] rather than Tables pre-calculated by LUIN. The program can calculate any of six different doses received on a flight following the shortest route between the origin and destination airports entered by the user. The dose endpoints are: Ionization, absorbed dose in silicon, absorbed dose in tissue, tissue dose equivalent, ambient dose equivalent H*(10), and Effective dose. The route followed is the geodesic calculated by computer programs INVERSE and FORWARD, maintained by the National Oceanic and Atmospheric Administration (NOAA).[18] These programs take into account the differences between the shape of the Earth and a perfect sphere. In addition to the origin and destination, the user also enters the flight profile, which consists of minutes to reach the first en route altitude from liftoff, altitude and minutes spent at each en route altitude, and minutes descending to the destination airport from the last en route altitude. When calculating a dose, the program takes into account the effects of changes in altitude, geographic coordinates, solar activity, and the Earth's magnetic field. The last two variables are accounted for in the NAIRAS Tables and are external to CARI.

Like CARI-6W and CARI-7, and unlike other previous versions of CARI, the core programming of CARI-NAIRAS is Fortran95.[19] Fortran95 compilers are readily available for both Microsoft Windows® operating systems and the vast proprietary and freeware Unix family of operating systems. Fortran is highly standardized, with vendors offering compilers with various non-standard extensions to the language, in addition to a fixed core following the standard set by a joint committee of the International Standards Organization and the International Electrotechnical Commission.[19] Thus, with minor modifications, CARI-NAIRAS should be useable on practically any system with sufficient disk space and memory, from laptop to mainframe. System requirements are expected to be less than 300 MB of free memory and 6 GB of free disk space, with almost all the disk space needed for the NAIRAS databases. A description of the source code and its organization is provided in Appendices B and C.

III. RESULTS

Table 2 lists effective doses for 32 domestic and international flights (see Appendix D for flight profile information), as calculated by CARI-6W, CARI-7, and CARI-NAIRAS using 1960 average solar activity conditions. The data show that flight dose estimates are similar, with all three programs typically estimating a dose within 20% of the mean of the three programs (CARI-6W, 3 > 20%; CARI-7, 2 > 20%; CARI-NAIRAS, 8 > 20%). CARI-NAIRAS estimates are expected to improve once the latest version of HZETRN is incorporated into the system.

Table 2. CARI-7 and CARI-NAIRAS effective doses calculated for selected flights using 1960 average solar activity conditions.

	Effective Dose (microSv/hr)		
Origin-Destination	**CARI-6W**	**CARI-7**	**CARI-NAIRAS**
HOUSTON, TX, USA - AUSTIN, TX, USA	1.44E-01	1.61E-01	3.51E-01
SEATTLE, WA, USA - PORTLAND, OR, USA	1.38E-01	1.51E-01	3.72E-01
MIAMI, FL, USA - TAMPA, FL, USA	3.06E-01	3.58E-01	6.24E-01
ST.LOUIS, MO, USA - TULSA, OK, USA	1.19E+00	1.16E+00	1.71E+00
TAMPA, FL, USA - ST.LOUIS, MO, USA	3.31E+00	3.36E+00	4.67E+00
SAN JUAN, PUERTO RICO - MIAMI, FL, USA	3.75E+00	3.90E+00	4.15E+00
DENVER, CO, USA - MINNEAPOLIS-ST.PAUL, MN, USA	2.69E+00	2.50E+00	3.60E+00
NEW ORLEANS, LA, USA - SAN ANTONIO, TX, USA	2.00E+00	2.18E+00	2.15E+00
NEW YORK, NY, USA - SAN JUAN, PUERTO RICO	6.99E+00	7.00E+00	8.14E+00
LOS ANGELES, CA, USA - HONOLULU, HI, USA	1.02E+01	1.01E+01	8.00E+00
CHICAGO, IL, USA - NEW YORK, NY, USA	4.70E+00	3.94E+00	5.43E+00
HONOLULU, HI, USA - LOS ANGELES, CA, USA	1.14E+01	1.11E+01	8.09E+00
WASHINGTON, DC, USA - LOS ANGELES, CA, USA	1.25E+01	1.21E+01	1.46E+01
TOKYO, JAPAN - LOS ANGELES, CA, USA	2.39E+01	2.20E+01	2.29E+01
MINNEAPOLIS-ST.PAUL, MN, USA - NEW YORK, NY, USA	5.77E+00	4.99E+00	6.76E+00
LONDON, UK - DALLAS, TX, USA	2.99E+01	2.64E+01	3.76E+01
NEW YORK, NY, USA - CHICAGO, IL, USA	6.31E+00	5.22E+00	6.88E+00
LOS ANGELES, CA, USA - TOKYO, JAPAN	2.95E+01	2.87E+01	2.92E+01
DALLAS, TX, USA - LONDON, UK	2.70E+01	2.36E+01	3.33E+01
LISBON, PORTUGAL - NEW YORK, NY, USA	1.96E+01	1.86E+01	2.28E+01
SEATTLE, WA, USA - ANCHORAGE, AK, USA	1.13E+01	9.75E+00	1.36E+01
CHICAGO, IL, USA - SAN FRANCISCO, CA, USA	1.32E+01	1.17E+01	1.35E+01
SEATTLE, WA, USA - WASHINGTON, DC, USA	1.51E+01	1.29E+01	1.66E+01
NEW YORK, NY, USA - SEATTLE, WA, USA	1.87E+01	1.58E+01	2.05E+01
LONDON, UK - NEW YORK, NY, USA	2.50E+01	2.16E+01	2.90E+01
SAN FRANCISCO, CA, USA - CHICAGO, IL, USA	1.39E+01	1.22E+01	1.40E+01
CHICAGO, IL, USA - LONDON, UK	2.92E+01	2.41E+01	3.25E+01
TOKYO, JAPAN - NEW YORK, NY, USA	4.76E+01	3.95E+01	5.00E+01
LONDON, UK - LOS ANGELES, CA, USA	4.20E+01	3.49E+01	4.57E+01
NEW YORK, NY, USA - TOKYO, JAPAN	5.10E+01	4.31E+01	5.28E+01
LONDON, UK - CHICAGO, IL, USA	3.22E+01	2.66E+01	3.56E+01
ATHENS, GREECE - NEW YORK, NY, USA	4.11E+01	3.53E+01	3.94E+01

Figure 1 shows the effective dose profile as calculated by CARI-7, which uses a database of cosmic ray showers calculated by Monte Carlo radiation transport code MCNPX2.7.0 (Monte Carlo N-Particle Transport Code for Multi-Particle and High-Energy Applications),[6] the Monte Carlo radiation transport code PHITS (Particle and Heavy Ion Transport code System),[20,21] and the NAIRAS model. For the PHITS calculations in Figure 1, heavy ion fluence-to-effective dose conversion coefficients are those calculated by Sato et al. (2003)[22] using ICRP Pub. 60 guidance. The coefficients for protons and neutrons were calculated by Sato et al. (2009)[23] using ICRP Pub. 103 recommendations, and the coefficients for other particles (muons, charged pions, electrons, positrons, and photons) were calculated using ICRP Pub. 60 recommendations.[10] The fluence-to-effective dose conversion coefficients used for the CARI-7 calculations are almost identical to those used by PHITS. They are based on the same organ dose data, but coefficients based on ICRP Pub. 60 guidance were recalculated using ICRP Pub. 103-recommended tissue weighting factors. In NAIRAS, the fluence-to-effective dose conversion coefficients are based upon the neutron and proton coefficients collected by Pelliccioni,[11] which were calculated using ICRP Pub. 60 guidance. For neutrons and protons, the coefficients are used directly; for heavier particles, the coefficients are scaled to the proton coefficients by (Zeff)2/A, where Zeff is the effective charge, which takes into account the electron capture by heavy ions at low energies. Despite the differences in transport methods and estimates of fluence-to-effective dose conversion coefficients, agreement of the three programs is very good at altitudes between 20 g.cm-2 (87,000 ft; FL 870) and 350 g.cm-2 (27,000 ft; FL 270), where most jet aircraft operate.

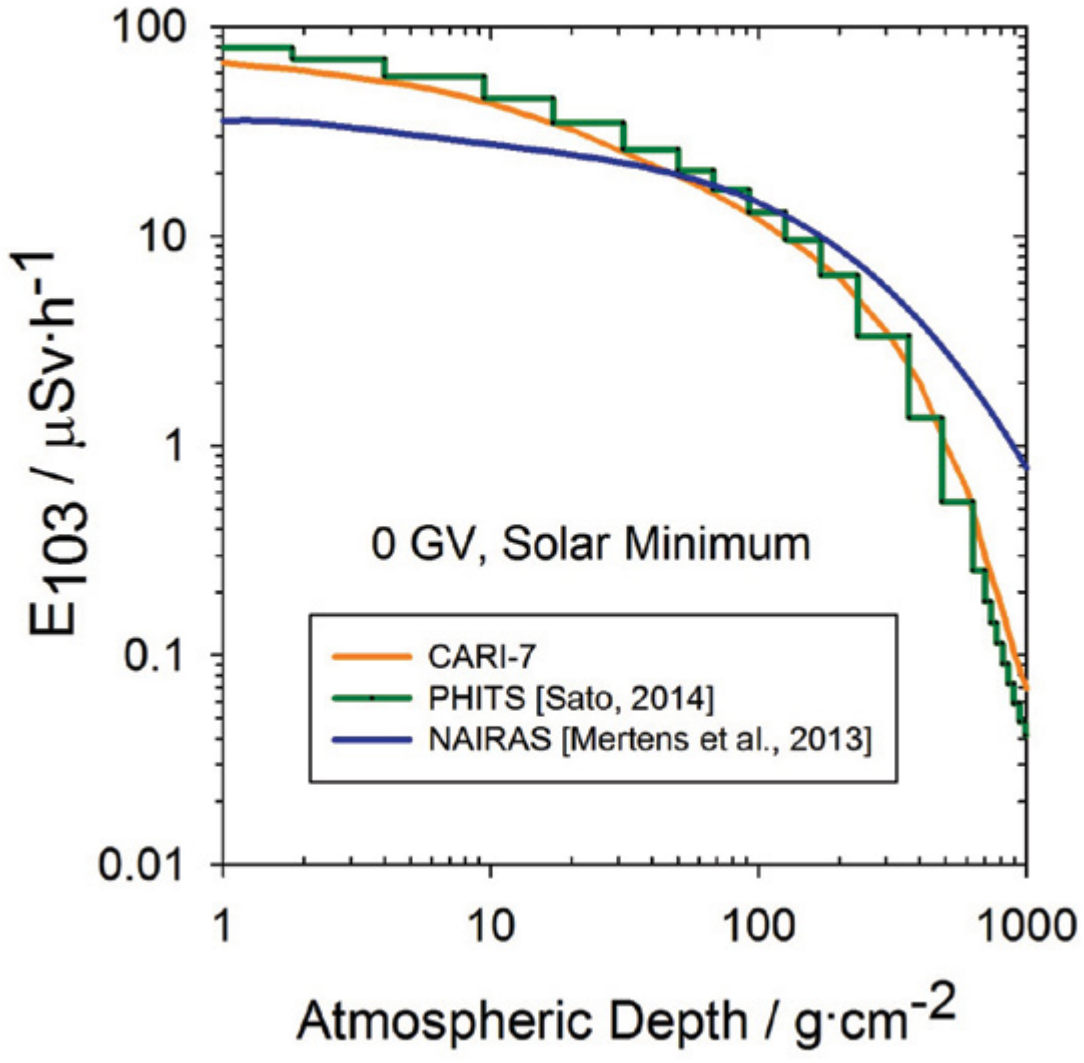

Figure 1. ICRP Pub. 103 effective dose rate versus atmospheric depth as calculated by CARI-7, PHITS, and NAIRAS.

IV. DISCUSSION

CARI-NAIRAS is a good estimator of flight dose at most commercial cruise altitudes. This is not surprising, since NAIRAS generally compares well with in-flight measurement data.[17] This comparison showed that NAIRAS was accurate for most conditions, but could use improvement, as demonstrated by the deviation from CARI-6W and CARI-7 on the lowest altitude flights. However, this shortcoming is temporary. The accuracy of NAIRAS, already good for most flights conditions, will improve as HZETRN continues to develop. The databases are easy to replace, program updates are as easy as adding new files to the database directories, and CARI-NAIRAS, unlike CARI-6, is nearly unlimited in flight altitude. It is a boon to both regular CARI users and NAIRAS users in that it brings the ease of the CARI flight dose calculation interface to those who wish to use NAIRAS to estimate doses on aircraft flights, as well as bringing CARI users the ability to use NAIRAS to estimate flight doses for manned commercial space flights and high-altitude manned balloon flights without concern for error introduced by the superposition approximation, which was present in all versions of CARI, up through CARI-6.

V. REFERENCES

1. O'Brien, K. *LUIN, a code for the calculation of cosmic ray propagation in the atmosphere.* USAEC Report HASL-275; 1973.

2. Schaefer, HJ. Radiation exposure in air travel. *Science*, 1971, 173, 780-783.

3. O'Brien, K; Friedberg, W. Atmospheric cosmic rays at aircraft altitudes. *Environ Intl* 20(5):645-663 (1994).

4. O'Brien, K; Friedberg, W. Atmospheric cosmic rays and solar energetic particles at aircraft altitudes. *Environ Int* 22(Suppl 1):S9-S44 (1996).

5. O'Brien, K; Smart, DF; Shea, MA; et al. World-wide radiation dosage calculations for air crew members. *Advan Space Res*, 2003, 31(4), 835-840.

6. Oak Ridge National Laboratory (ORNL). *Monte Carlo N-particle transport code system for multiparticle and high energy applications (MCNPX 2.7.0)*, RSICC code package C740, developed at Los Alamos National Laboratory, released 2011 (available from the Radiation Safety Information Computational Center at ORNL, Oak Ridge, TN).

7. Wilson, JW; Townsend, LW; Schimmerling,W; et al. *Transport methods and interactions for space radiation.* NASA Report RP-1257, 1991.

8. Wilson, JW; Badavi, FF; Cucinotta, FA; et al. *HZETRN: Description of a free-space ion and nucleon transport and shielding computer program.* NASA Technical Paper No. 3495. Hampton, VA: National Aeronautics and Space Administration, Langley Research Center; 1995.

9. Prof. O'Brien can be contacted through Northern Arizona University e-mail: Keran.O'Brien@nau.edu.

10. International Commission on Radiological Protection (ICRP). *1990 Recommendations of the International Commission on Radiological Protection*. ICRP Publication 60. Tarrytown, NY: Elsevier Science; 1991.

11. Pellicioni, M. Overview of fluence-to-effective dose and fluence-to-ambient dose equivalent conversion coefficients for high energy radiation calculated using the FLUKA code. *Radiat Prot Dosim*, 2000, 88(4), 279-297.

12. Copeland, K. *Recent and Planned Developments in the CARI Program*. FAA Office of Aerospace Medicine Report No. DOT/FAA/AM-13/6. Washington, DC: Office of Aerospace Medicine; 2013.

13. Copeland, K.A. *Cosmic ray particle fluences in the atmosphere resulting from primary cosmic ray heavy ions and their resulting effects on dose rates to aircraft occupants as calculated with MCNPX 2.7.0.* A doctoral thesis at the Royal Military College of Canada (RMC): available from the author on request or in electronic format from the Massey Library collection at RMC and from the Canadian National Archives in Ottawa, Canada. Accepted July 2014.

14. International Commission on Radiation Units and Measurements (ICRU). *Quantities and units in radiation protection dosimetry*. ICRU Report 51. Bethesda, MD, USA: ICRU Publications; 1993.

15. International Commission on Radiological Protection (ICRP). *The 2007 recommendations of the International Commission on Radiological Protection*. ICRP Publication 103. London, UK: Elsevier; 2007.

16. International Commission on Radiological Protection (ICRP). *Adult reference computational phantoms.* ICRP Publication 110. Oxford, UK: Pergamon; 2009.

17. Mertens, CJ; Meier MM; Brown, S; et al. NAIRAS aircraft radiation model development, dose climatology, and initial validation. *Space Weather*, 2013, 11, 1-33, doi:10.1002/swe.20100.

18. Frakes, SJ. Computer programs INVERSE and FORWARD. 2002. Available at: www.ngs.noaa.gov/PC_PROD/Inv_Fwd/; last visited 6/29/2012. (Adapted for CARI-NAIRAS as GEODESIC.FOR).

19. Fortran Standards Working Group, IEC ISO JTC1/SC22/WG5. *Welcome to the official home of Fortran Standards*. http://www.nag.co.uk/sc22wg5/; last visited 6/29/2012.

20. Sato, T; Private communication; 2014.

21. Sato,T; Niita, K. Analytical functions to predict cosmic-ray neutron spectra in the atmosphere. *Radiat Res*, 2006, 166, 544-555.

22. Sato, T; Tsuda, S; Sakamoto, Y; et al. Analysis of dose-LET distribution in the human body irradiated by high energy hadrons. *Radiat Prot Dosim,* 2003, 106, 145-153.

23. Sato, T; Endo, A; Zankl, M; et al. Fluence-to-dose conversion coefficients for neutrons and protons calculated using the PHITS code and ICRU/ICRP adult reference computational phantoms. *Phys Med Biol,* 2009, 54, 1997-2014, doi: 10.1088/0031-9155/54/7/009.

24. O'Neill PM. Badhwar–O'Neill 2010 galactic cosmic ray flux model—Revised. *IEEE Transactions on Nuclear Science*, 2010 (Dec), 57(6), 3148-3153, doi: 10.1109/TNS.2010.2083688.

25. Kress, BT; Mertens CJ; Wiltberger, M. Solar energetic particle cutoff variations during the 29-31 October 2003 geomagnetic storm. *Space Weather*, 2010, 8, S05001, doi: 10.1029/2009SW000488.

26. Mertens, CJ; Kress, BT; Wiltberger, M; et al. Geomagnetic influence on aircraft radiation exposure during an energetic solar particle event in October 2003. *Space Weather*, 2010, 8, S03006, doi:10.1029/2009SW000487.

27. Tsyganenko NA; Sitnov, NI. Modeling the dynamics of the inner magnetosphere during strong geomagnetic storms. *J Geophys Res*, 2005, 110, A12108, doi:10.1029/2005JA011250.

28. Picone, JM; Hedin, AE; Drob, DP; Aikin, AC. NRLMSIS-00 empirical model of the atmosphere: statistical comparisons and scientific issues. *J Geophys Res*, 2002, 107(A12), 1468 (SIA 15-1 - SIA 15-16).

29. Duey, A. Computer module Qsort, 2004. Available at: www.andrewduey.com/cscs252d.htm; last visited 30/05/2012 **(a**dapted for character arrays as Shellsort and Shellsort2 for CARI-NAIRAS).

APPENDIX A.
NAIRAS Summary

The U.S. National Aeronautics and Space Administration (NASA) operates a near real-time, global, physics-based, data-driven model for the prediction of biologically hazardous atmospheric radiation exposure. The model is called Nowcast of Atmospheric Ionizing Radiation for Aviation Safety (NAIRAS).(17) Graphical and tabular data products from the operational prototype are streaming live from the NAIRAS public website at: http://sol.spacenvironment.net/~nairas/

A subset of the NAIRAS real-time graphical products is available on the SpaceWx smartphone app for iPhone, iPad, and Android. The NAIRAS model provides data-driven, global, real-time predictions of atmospheric ionizing radiation exposure rates on a geographic 1° by 1° latitude and longitude grid from the surface of the Earth to 90 km, with a vertical resolution of 1 km. The real-time, global predictions are updated every hour. The developers of NAIRAS have adopted, as far as possible, the meteorological weather forecasting paradigm of combining physics-based forecast models with data assimilation techniques. Physics-based models are utilized within NAIRAS to transport cosmic rays through three distinct zones: the heliosphere, Earth's magnetosphere, and the neutral atmosphere. As much as possible, real-time measurement data are used to both specify the ionizing radiation field at the boundaries of the zones and characterize the internal properties of each zone. Thus, they provide observational constraints on the physics-based models that improve the simulations of transport and transmutation of galactic cosmic radiation (GCR) and solar cosmic radiation (SCR) through the three zones.

Primary GCR are transported through the heliosphere to the vicinity of the Earth using an expanded version of the 2010 update of the Badhwar and O'Neill GCR model,(24) which uses ground-based neutron monitor count rate measurements from the Climax neutron monitor site in order to provide a measurement constraint on the simulated solar cycle modulation of the GCR spectrum at 1 AU. This enables accurate predictions of GCR spectra, at least on monthly to seasonal time scales. The NAIRAS team has extended the application of neutron monitor data by incorporating four high-latitude neutron monitor count rate measurements into the GCR model predictions at 1 AU. The additional neutron monitor stations are Thule (Greenland), Oulu (Finland), Izmiran (or Moscow) (Russia), and Lomnicky Stit (Slovakia). The reasons for utilizing these neutron monitor data are two-fold: (1) high-latitude locations are sensitive to the GCR spectral region most influenced by solar cycle variability, and (2) data from these stations are available in real-time or near real-time. Also, the solar modulation parameter derived from Climax (U.S.A.) neutron count rates has been recently extended from 1958-2009. This extended Climax-based solar modulation parameter provides the reference solar modulation parameter from which to derive a real-time GCR model suitable for integration into the NAIRAS model. The NAIRAS GCR model was developed by cross-correlating the Climax-based solar modulation parameter with the neutron count rates measured at the four high-latitude sites mentioned above.

The minimum access energy to the neutral atmosphere is determined based on the cutoff rigidity for each incident charged particle. NAIRAS real-time geomagnetic cutoff rigidities are computed from numerical solutions of charged particle trajectories in a dynamically varying geomagnetic field that includes both the internal magnetic field and the magnetospheric magnetic field contributions.(25,26) The cutoff rigidity code was developed by the Center for Integrated Space Weather Modeling (CISM) at Dartmouth College. In particular, the specification of the geomagnetic field due to Earth's internal field source is provided by the internal geomagnetic reference field (IGRF) model, while the real-time dynamical response of the magnetospheric magnetic field to solar wind conditions and interplanetary magnetic field is provided by the TS05 model.(27) While other model selections are available, at present, the simulated real-time geomagnetic cutoff rigidities are calculated with the TS05 model, using the IGRF model for comparison.

The NAIRAS model uses the physics-based deterministic transport code HZETRN (High Charge (Z) and Energy TRaNsport) to transport cosmic rays through the atmosphere.(8) The HZETRN transport calculations are continuously updated using real-time measurements of boundary condition specifications of the space radiation environment and of atmospheric density versus altitude; GCR and solar energetic particle (SEP) atmospheric radiation exposure predictions are both included in real-time. In the NAIRAS model, there are 59 coupled transport equations in the HZETRN description of GCR transport through the atmosphere. This set includes transport equations for neutrons and GCR nuclear isotopes from protons through nickel (Z=28, A=58). NCAR/NCEP pressure versus geopotential height data is extended in altitude above 10 hPa using the Naval Research Laboratory Mass Spectrometer and Incoherent Scatter (NRLMSIS) model atmosphere.(28) NCAR/NCEP and NRLMSIS temperatures are smoothly merged at 10 hPa at each horizontal grid point. NRLMSIS temperatures are produced at 2 km vertical spacing from the altitude of the NCEP/NCAR 10 hPa pressure surface to approximately 100 km. The pressure at these extended altitudes can be determined from the barometric law using the NRLMSIS temperature profile and the known NCAR/NCEP 10 hPa pressure level, which assumes the atmosphere is in hydrostatic equilibrium and obeys the ideal gas law. Finally, the altitudes and temperatures are linearly interpolated in log pressure to a fixed pressure grid from 1000 hPa to 0.001 hPa, with six pressure levels per decade. The result from this step is pressure versus altitude at each horizontal grid point from the surface to approximately 100 km. Atmospheric depth ($g \cdot cm^{-2}$) at each altitude level and horizontal grid point is computed by vertically integrating the mass density from a given altitude to the top of the atmosphere. The mass density is determined by the ideal gas law using the pressure and temperature at each altitude level. The result from this step produces a 3-D gridded field of atmospheric depth. Atmospheric depth at any specified aircraft altitude is determined by linear interpolation along the vertical grid axis in log atmospheric depth.

APPENDIX B.
CARI-NAIRAS Code Description

For a complete listing of the FAA-funded source code, see Appendix C. Descriptions of the major functions are in Table B.1. Subroutines are described in Table B.2.

Table B.1. Functions

Function	Purpose
ALTCODE	Finds alternates to ICAO codes in the file CODES.
BADSTR	Handles response to errors in BIG files.
DOSTR	Used to label output of dose units.
DRSTR	Used to label output of dose rates units.
FDR	The top level routine called for finding the dose rate. Inputs are date, latitude, longitude, altitude, and dose type.
GTQ	A logical function that evaluates the strings lexically to decide which comes first for subroutines SHELLSORT and SHELLSORT2.
SPLINE_4PT	Implements a 4-point Catmul-Rom cubic spline, assuming that the data points are evenly spaced in 1-d. It is used for interpolation on the latitude grid, longitude grid, and altitude grid.

Table B.2. Subroutines

Subroutine	Purpose
ADDAPORT, (AIRPORT_MENU, FINDPORT, LOCATIONS, MAINMENU, OUTPUT_MENU, RUNBIG)	These subs provide text for the various menus the user encounters in the program.
ALTNOW	Finds the aircraft altitude at any time during a flight.
BIG_FLT_DOSE	Calculates a flight dose based upon the information supplied about the flight profile by RDBIGFLT and any user entered overrides such as a new flight date.
CLS	Mimics the CLS command from DOS by the method indicated in CARI.INI environment variable 'CLS' as 'DOS' (the default) or 'SCROLL.'
DATE2YMD	Converts the date from a string to a set of integer values for use by other subroutines and functions.
EPITATH	Writes a final message to the user in the event of a serious error and ends the program.
GETINI	Controls the reading of the .INI file, which contains the variables available to users to set externally.
LAT_BRACKETS (LON_BRACKETS, ALT_BRACKETS)	Finds the surrounding four latitudes (longitudes, altitudes) to use in SPLINE_4PT. Since the world is round, but represented by a sheet, longitudes near the prime meridian use points from the other end of the sheet, as if it were a cylinder with 360^{o} and 0^{o} being the same.
LC2UC	Converts lower case ASCII characters to upper case ASCII characters.
LOADER73	Reads the user-selected NAIRAS table into memory. All the dose rates at points (x=1, y=1, z=1) and (x=2, y=2, z=7) are printed to the diagnostic file to allow the user to easily confirm the proper table was loaded.

Table B.2. (Continued)

Subroutine	Purpose
MAKE_NDX	Reads the permanent and user entered airport databases, joins them and calls sorting routines (SHELLSORT, SHELLSORT2) to sort the contents by city name and airport name. The resulting sorted databases are called CITY.NDX and PORT.NDX, respectively. If airports with identical codes are in both AIRPORTS.DAT and NEWPORTS.DAT, the data for the airport in NEWPORTS.DAT are used. If there are multiple airports with the same code in NEWPORTS.DAT, the first airport with the right code is used.
MENUHEADER	Controls the text in the header above each of the menus in the command prompt display box.
ONESHOT	A master subroutine for a user seeking to find the dose rate at a single point in time and space. It is subdivided into three parts: data input, calculating the dose rate, and reporting the results.
OOPS	Writes a message to the user in the event of a minor error, then returns the user to the most recent menu.
OPENDATABASES	Loads airport databases and initializes the diagnostic output file.
RDBIGFLT	Reads flight profiles from a *.BIG file and provides the flight information to BIG_FLT_DOSE.
READKB	A data input routine to replace the READ intrinsic function for user entered data from the keyboard. For complex entries of more than one word, it provides the user with input instructions. It then calls READ and then converts all user-entered lower case characters to uppercase characters by calling LC2UC.
RUNBIG	Controls the evaluation of big file data.
SHOWPICK	Opens the indicated file using the file opener specified in the CARI.INI file by the environment variable 'VIEWER'. The command 'VIEWER filename' (e.g., 'notepad FL32.BIG') is sent to the system. The default viewer is Microsoft's Notepad, which ships with the Windows operating system. Any installed text editor (e.g., EDIT or EMACS) which can be run with the command structured as noted could be used. The user need only change the CARI.INI file.
USE_FORWARD and USE_INVERSE	Control use of the subroutine collection in GEODESIC.FOR adapted from NOAA's FORWARD.FOR and INVERSE.FOR.

APPENDIX C.
Source Code for CARI-NAIRAS (as of 26 Sep. 2014)

It is important to note that not all of the code used in CARI-NAIRAS was written at CAMI. The following source code contains only code written at CAMI. For those wishing to obtain a complete listing of the source code, other codes should be obtained from their sources.[18,29]

```
!
!  Program CARI-NARIAS
!
!  Fortran coding by Kyle Copeland, Civil Aerospace Medical Institute
!  Coded to have the look and feel of the CARI-6 QuickBASIC Code set
!  mostly coded by Frances Duke, Lo Snyder, Wallace Friedberg and Kyle
!  Copeland.
!   (See Help file for complete list of developers of CARI-6)
!
!
  PROGRAM CN
  ! CARI-NAIRAS, CARI that runs from NAIRAS tables
  IMPLICIT NONE
   CHARACTER(10)::INIVAR
   CHARACTER(12)::INIVAL
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE
   CHARACTER(1)::NEWFORMAT
   INTEGER::DTEDIM1
  COMMON/INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
  COMMON/NAIRAS/NEWFORMAT,DTEDIM1

   CALL GETINI(MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT)
   PRINT*, 'DISPLAY MENUS = ',MENUS
   PRINT*, 'OS = ',OS
   PRINT*, 'DISPLAY = ',DISPLAY
   PRINT*, 'DIAGNOSTIC PRINTING = ',DIAGNOSE
   PRINT*, 'VIEWER = ',VIEWER
   PRINT*, 'OUPUT = ',OUTPUT
   CALL OPENDATABASES
        IF (MENUS.EQ.'NO!') THEN
   WRITE (*,*) 'Starting analysis of DEFAULT.BIG profiles'
              CALL RUNBIG
   WRITE (*,*) 'Finished analysis of DEFAULT.BIG profiles'
   WRITE (*,*) 'Results in DEFAULT.OUT'
   STOP
   ENDIF
   CALL MAINMENU


        END PROGRAM CN
! END OF PROGRAM CARI MAIN MODULE
!
! Include source code for geodesics
! These code are freely available from NOAA
! These subs use double precision i/o
!
  INCLUDE 'GEODESIC.FOR'
!    inverse(in:: lat1, lon1, lat2, lon2
!      out:: faz, baz, meters
!    forward(in:: lat1, lon1, faz, meters
!      out:: lat2, lon2
!                7
!----6-------------------------------------------------------------------2
! START OF FUNCTIONS AND SUBS TO GET DOSE RATE
! Written by Kyle Copeland
  SUBROUTINE OPENDATABASES
! OPEN OTHER PERMANENT DATABASES

  OPEN(UNIT=29,FILE='FT-GM.DAT',STATUS='OLD')
  OPEN(UNIT=30,FILE='AIRPORTS\CODES',STATUS='OLD')
! OPEN TRANSIENT DATABASES
```

```
  OPEN(UNIT=40, FILE='diagnose\DIAGNOSE.DAT',STATUS='UNKNOWN')
! OTHER UNITS USED IN THE PROGRAM
!  UNIT 21, *.EV*
!  UNIT 19, DATAIN
!  UNIT 20, DATAOUT
!  UNIT 96, INVERSE AND FORWARD FILES
!  UNIT 99, SCRATCH
!
  ! The number of airports is machine dependent
  ! and can lead to stack space errors, >5000 seems typical
  CALL MAKE_NDX !WORKS WITH 5600, BUT 5700 TOO MANY
  ! CALL MAKE_NDX(91,5200) this sort of works,
  ! Some airports not read correctly
  ! Use increased stack size to allow larger numbers
  END SUBROUTINE OPENDATABASES
!                 7
!----6---------------------------------------------------------------2
  SUBROUTINE LOADER73(DTE)
! This reads 'new format' NAIRAS data files
  IMPLICIT NONE
  CHARACTER(72):: JUNK
  REAL, DIMENSION(96,144,9,91):: DTE
  INTEGER::i,j,k,l
  CHARACTER(1)::NEWFORMAT
  INTEGER::DTEDIM1, J1, J2
  COMMON/NAIRAS/NEWFORMAT,DTEDIM1

  WRITE (40,*)'ACCUMULATING ANNUAL DATA'
  READ (21,*) JUNK
  READ (21,93) DTEDIM1, J1, J2
  IF (DTEDIM1.eq.96) THEN
  NEWFORMAT='N'
  WRITE(40,*)'DATA ARE IN OLD 96-LON X 8 TABLE FORMAT'
  ELSE IF (DTEDIM1.eq.73) THEN
  NEWFORMAT='Y'
  WRITE(40,*)'DATA ARE IN NEW 73 LON X 9 TABLE FORMAT'
  ELSE !SOMETHING IS WRONG
  CLOSE(21)
  WRITE(40,*)'NAIRAS DATA ARE IN UNKNOWN FORMAT'
  CALL EPITATH('UNABLE TO READ NAIRAS DATA, UNEXPECTED FORMAT',45)
  ENDIF
  DO i=1,12
  READ (21,*) JUNK
  ENDDO
  DO i=1,DTEDIM1
  DO j = 1,144
   READ (21,*) JUNK
   READ (21,*) JUNK
   IF (NEWFORMAT.EQ.'Y') THEN
   DO k = 1,9
   READ (21,*) JUNK
   DO l=1,85,6
    READ(21,92) DTE(i,j,k,l),DTE(i,j,k,l+1),DTE(i,j,k,l+2), &
  &     DTE(i,j,k,l+3),DTE(i,j,k,l+4),DTE(i,j,k,l+5)
   ENDDO
   READ(21,95) DTE(i,j,k,91)
   ENDDO
   ELSE
   DO k = 1,9
   IF (k.EQ.5) GOTO 91 !MUST SKIP, NO SILICON IN OLD FORMAT
   READ (21,*) JUNK
   DO l=1,85,6
    READ(21,92) DTE(i,j,k,l),DTE(i,j,k,l+1),DTE(i,j,k,l+2), &
  &     DTE(i,j,k,l+3),DTE(i,j,k,l+4),DTE(i,j,k,l+5)
   ENDDO
   READ(21,95) DTE(i,j,k,91)
91   CONTINUE
   ENDDO
   ENDIF
  ENDDO
  ENDDO
  WRITE(40,*) 'Diagnostic gridpoints 1,1,4-9,1'
```

```
  WRITE(40,*) DTE(1,1,4,1)
  WRITE(40,*) DTE(1,1,5,1)
  WRITE(40,*) DTE(1,1,6,1)
  WRITE(40,*) DTE(1,1,7,1)
  WRITE(40,*) DTE(1,1,8,1)
  WRITE(40,*) DTE(1,1,9,1)
  WRITE(40,*) '...'
  WRITE(40,*) 'Diagnostic gridpoints 2,2,4-9,7'
  WRITE(40,*) DTE(2,2,4,7)
  WRITE(40,*) DTE(2,2,5,7)
  WRITE(40,*) DTE(2,2,6,7)
  WRITE(40,*) DTE(2,2,7,7)
  WRITE(40,*) DTE(2,2,8,7)
  WRITE(40,*) DTE(2,2,9,7)
  WRITE(40,*) '...'
  WRITE(40,*) 'ANNUAL DATA ACCUMULATION COMPLETE'
92 FORMAT(6ES12.4)
93 FORMAT(3I5)
95 FORMAT(1ES12.4)
  CLOSE(21)
  END SUBROUTINE LOADER73
!0-9999
!                 7
!----6-------------------------------------------------------------------2
  FUNCTION LINTERP(X1,X2,Y1,Y2,X)
!
! GENERAL LINEAR INTERPOLATION OF A Y-VALUE ASSOCIATED WITH AN X VALUE
! BETWEEN 2 KNOWN X,Y, PAIRS
   REAL::S,B,X,X1,X2,Y1,Y2,LINTERP
   CHARACTER(3)::DIAGNOSE='NO '
              !
   IF (X.EQ.X1) THEN
   LINTERP=Y1
   ELSEIF (X.EQ.X2) THEN
   LINTERP=Y2
   ELSE
   S=(Y2-Y1)/(X2-X1)
   B=Y1-S*X1
   LINTERP=S*X+B
   ENDIF
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'LINTERP IO'
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) X1,X2,Y1,Y2,X, ' Y=',LINTERP
  END FUNCTION LINTERP
!                 7
!----6-------------------------------------------------------------------2
  FUNCTION FDR(LAT,LON,KM,DT,DTE)
!
! FUNCTION TO FIND DOSE RATE OF TYPE DT AT ALTITUDE KM,
! AT LONGITUDE LON, AT LATITUDE LAT BASED ON DATA IN ARRAY DTE
!
!
!    TABLE=1, Atmospheric Pressure (kPa)
!    TABLE=2, Atmospheric Depth (g/cm**2)
!    TABLE=3, Atmospheric Density (g/cm**3)
!    TABLE=4, Atmospheric Ionization Rate (cm^-3 s^-1)
!    TABLE=5, Silicon Absorbed Dose Rate (uGy/hr)
!    TABLE=6, Tissue Absorbed Dose Rate (uGy/hr)
!    TABLE=7, Tissue Dose Equivalent Rate (uSv/hr)
!    TABLE=8, Ambient Dose Equivalent Rate (uSv/hr)
!    TABLE=9, Effective Dose Rate (uSv/hr)
!
!    DT=1, IONIZATION
!    DT=2, Si ABSORBDED DOSE RATE
!    DT=3, TISSUE ABSORBED DOSE RATE
!    DT=4, TISSUE DOSE EQUIVALENT RATE
!    DT=5, ICRU AMBIENT DOSE EQUIVALENT RATE, H*(10)
!    DT=6, EFFECTIVE DOSE RATE
!
! There are 73 (OR 96) latitudes, 144 longitudes, 91 altitudes, 5 dose types in 9 tables
  REAL, DIMENSION(96,144,9,91):: DTE
!
  REAL :: D64(4,4,4),DA(4,4), DAP(4)
```

```
  REAL, DIMENSION(4) :: DRALT
  INTEGER, DIMENSION(4) :: LONGRID,LATGRID,ALTGRID
  REAL :: LAT, LON, KM
  REAL :: DR
 ! DECLARE CALLED FUNCTIONS
  REAL :: LINTERP, SPLINE_4PT

  INTEGER :: RAD, DT, YEAR, TABLE
  INTEGER :: IALT, JALT, KALT, LALT
  INTEGER :: ILON, JLON, KLON, LLON, ILAT, JLAT, KLAT, LLAT

  CHARACTER(3)::DIAGNOSE='NO '
  FDR= 0.0
!
!STEP 1. FIND NEEDED (lat,lon) GRIDPOINTS FOR SPLINES
!
  WRITE(40,192) 'ECHO OF INCOMING DATA:',LAT,LON,KM,DT
192 FORMAT(A22,3F9.4,I3)
  WRITE (40,*) 'Getting lat and lon brackets'
  CALL LON_BRACKETS(LON, ILON, JLON, KLON, LLON)
  CALL LAT_BRACKETS(LAT, ILAT, JLAT, KLAT, LLAT)
! FILL VECTORS FOR EASY PROCESSING
  LONGRID(1) = ILON
  LONGRID(2) = JLON
  LONGRID(3) = KLON
  LONGRID(4) = LLON
  LATGRID(1) = ILAT
  LATGRID(2) = JLAT
  LATGRID(3) = KLAT
  LATGRID(4) = LLAT
  WRITE (40,*) 'Getting altitude brackets'
  CALL ALT_BRACKETS(KM,IALT,JALT,KALT,LALT)
  ALTGRID(1) = IALT
  ALTGRID(2) = JALT
  ALTGRID(3) = KALT
  ALTGRID(4) = LALT
!
!STEP 2. PERFORM 3 SUCCESSIVE SETS OF CUBIC SPLINE INTERPOLATIONS
!  (THIS REQUIRES 64 DOSE RATES!)
!
! USE CORRECT DOSERATE DATA TABLE
  TABLE = DT+3

! GET DOSES, INTERPOLATE
  DO I=1,4 !LATITUDE
  DO K = 1,4 !LONGITUDE
   DO L = 1,4 !ALTITUDE, look up dose rates at L
   D64(I,K,L) = DTE(LATGRID(I),LONGRID(K),TABLE,ALTGRID(L))
   WRITE(40,222) 'Dose ',TABLE-3,' rate at gridpoint ',I,K,L, &
  &' (',LATGRID(I),LONGRID(K),ALTGRID(L),' ) is ',D64(I,K,L) !diagnostic
   ENDDO
   DA(I,K) = SPLINE_4PT(1.,D64(I,K,1),D64(I,K,2),D64(I,K,3),  &
  &    D64(I,K,4),KM)
   WRITE(40,*)'Dose ',TABLE-3,' rate at gridpoint',i,k,' is ' &
  &,DA(I,K) !diagnostic
  ENDDO
!  INTERPOLATIONS IN LONGITUDE
  DAP(I) = SPLINE_4PT(2.5,DA(I,1),DA(I,2),DA(I,3),DA(I,4),LON)
  WRITE(40,*)'Dose ',TABLE-3,' rate at gridpoint',i,' is ',DAP(I) !diagnostic
  ENDDO
!  INTERPOLATIONS IN LATITUDE)
  FDR = SPLINE_4PT(2.5,DAP(1),DAP(2),DAP(3),DAP(4),LAT)

  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) FDR
222 FORMAT(A5,I2,A19,3I4,A3,3I4,A6,ES12.4)
  END FUNCTION FDR
!                 7
!----6---------------------------------------------------------------2
  FUNCTION CHAR2REAL(CHAR_IN,D)
! CAHR2REAL CONVERTS SHORT STRINGS TO REAL NUMBERS
!
   INTEGER :: D
```

```
            REAL :: CHAR2REAL
   CHARACTER(D)::CHAR_IN

   WRITE(40,*) 'Converting', CHAR_IN, 'to real'
            IF (CHAR_IN.EQ.'') THEN
            CHAR2REAL=0.0
   RETURN
   ENDIF
   OPEN (UNIT=99,STATUS='SCRATCH')
   WRITE(99,*) CHAR_IN
   REWIND 99
   READ(99,199) ZED
   CLOSE(99)
   CHAR2REAL=ZED
199           FORMAT (G16.7E2)
  END FUNCTION CHAR2REAL
!                 7
!----6-----------------------------------------------------------------2
  FUNCTION CHAR2INT(CHAR_IN,D)
!
! CHAR2INT CONVERTS SHORT STRINGS TO INTEGERS
!
   INTEGER::CHAR2INT,I,D
        CHARACTER(D)::CHAR_IN
   CHARACTER(3)::DIAGNOSE='NO '

!   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'CHAR_IN= ', CHAR_IN
            IF (TRIM(CHAR_IN).EQ.'') THEN
            CHAR2INT=0
   RETURN
   ENDIF
            DO I=1,D
                  IF (CHAR_IN(I:I).EQ.'-') THEN
                   CHAR_IN=CHAR_IN(I:D)
                        EXIT
                  ENDIF
   ENDDO
        OPEN (UNIT=99,STATUS='SCRATCH')
   WRITE(99,*) CHAR_IN
   REWIND 99
   READ(99,299) I
   CLOSE(99)
   CHAR2INT=I
299  FORMAT (I8)
        END FUNCTION CHAR2INT
!                 7
!----6-----------------------------------------------------------------2
  FUNCTION SPLINE_4PT(DELTA, VI, VJ, VK, VL, X)
!
! A CATMUL-ROM CUBIC SPLINE
! A CUBIC SPLINE REQUIRING 4 DATA POINTS, 1 ABOVE X, 2 BRACKETING X,
! AND 1 BELOW X. ASSUMES 0<=x<1, EVENLY SPACED POINTS ON X AXIS
! B AND C ARE BRACKETING X COORDS, VA-VD ARE VALUES OF F(X) AT
! POINTS A-D, XN IS NORMALIZED X
!
  INTEGER :: J
  REAL :: DELTA, X, XN, VI, VJ, VK, VL, F, A0, A1, A2, A3
       REAL :: SPLINE_4PT
  CHARACTER(3) :: DIAGNOSE='NO!'
!
! 1. NORMALIZE X TO A VALUE FROM 0 TO 1 [INCLUSIVE]
!  DELTA = DISTANCE BETWEEN POINTS
  XN = MOD(X,DELTA)/DELTA
!  WRITE(40,*) 'XN=',XN,' X=',X,' DELTA=',DELTA
       !
! 2. CALCULATE F(X)
  A0 = -0.5*VI +1.5*VJ -1.5*VK+0.5* VL
  A1 = VI - 2.5*VJ + 2.0*VK - 0.5*VL
  A2 = 0.5*VK - 0.5*VI
  A3 = VJ
  F= A0*XN**3+A1*XN**2+A2*XN+A3
  SPLINE_4PT = F
```

```
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'SPLINE INPUT', VI,VJ,VK,VL,X
  IF (DIAGNOSE.EQ.'YES') WRITE(*,*) 'SPLINE INPUT', VI,VJ,VK,VL,X
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'COEFFICIENTS', A0,A1,A2,A3
  IF (DIAGNOSE.EQ.'YES') WRITE(*,*) 'COEFFICIENTS', A0,A1,A2,A3
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'SPLINE STEP AND OUTPUT'  &
  & ,DELTA, F
  IF (DIAGNOSE.EQ.'YES') WRITE(*,*) 'SPLINE STEP AND OUTPUT'  &
  & ,DELTA, F
  END FUNCTION SPLINE_4PT
!                 7
!----6----------------------------------------------------------------2
  SUBROUTINE LON_BRACKETS(XLON, I, J, K, L)
!
!  FINDS I-L VALUES FOR SPLINE_4PT FROM LONGITUDE GRID BASED ON
!   XLON
!  values of xlon: -180<=xlon<=180
!
  REAL :: XLON, XN, TEMP
  INTEGER :: IXN, I, J, K, L
!
!  range of allowed values for index is 1-144,
!
  IF (XLON.GE.0.) THEN
   TEMP = XLON
  ELSE
   TEMP= 360.+ XLON
!  180 E - 357.5 E (180 W - 2.5 W)
  ENDIF
  XN = TEMP/2.5
  IXN = INT(XN)+1 ! ROUND TO NEAREST INTEGER
!  grid pt 1 is 0 deg E , grid point 73 is 180 E (180 W)
  J=IXN
  SELECT CASE (J)
  CASE (1)
       I = 144
   K = 2
   L = 3
  CASE (143)
   I = 142
   K = 144
   L = 1
  CASE (144)
   I = 143
   K = 1
   L = 2
  CASE DEFAULT
   I = J-1
   K = J+1
   L = J+2
  END SELECT

  END SUBROUTINE LON_BRACKETS
!                 7
!----6----------------------------------------------------------------2
  SUBROUTINE LAT_BRACKETS(XLAT, I, J, K, L)
!
!  FINDS I-L VALUES FOR SPLINE_4PT FROM LATITUDE GRID BASED ON
!   XLAT
!  step size for data is 2.5 degrees, starts at 90 north
!
  IMPLICIT NONE
  REAL :: XLAT, XN
  INTEGER :: IXN, I, J, K, L
  CHARACTER(1) :: NEWFORMAT
  INTEGER :: DTEDIM1
  COMMON/NAIRAS/NEWFORMAT,DTEDIM1
!
  IF (NEWFORMAT.EQ.'Y')THEN
  XN = XLAT/2.5
  IXN = -1*INT(XN)+37 ! ROUND TO NEAREST POSITIVE INTEGER 1-73
  SELECT CASE(IXN)
  CASE(1) !at or close to North Pole
```

```
        I = IXN
    J = IXN
    K = IXN+1
    L = IXN+1
   CASE(72)! Close to South Pole
        I = IXN
    J = IXN
    K = IXN+1
    L = IXN+1
   CASE(73) !at South pole
        I = IXN
    J = IXN
    K = IXN-1
    L = IXN-1
   CASE default !away from any poles
        I = IXN-1
    J = IXN
    K = IXN+1
    L = IXN+2
   END SELECT
   ELSE! OLD FORMAT
   XN = XLAT/1.89474
   IXN = -1*INT(XN)+48 ! ROUND TO NEAREST POSITIVE INTEGER 1-96
   SELECT CASE(IXN)
   CASE(1) !at or close to North Pole
        I = IXN
    J = IXN
    K = IXN+1
    L = IXN+1
   CASE(95) !Close to South Pole
        I = IXN
    J = IXN
    K = IXN+1
    L = IXN+1
   CASE(96) !at South pole
        I = IXN
    J = IXN
    K = IXN-1
    L = IXN-1
   CASE default !away from any poles
        I = IXN-1
    J = IXN
    K = IXN+1
    L = IXN+2
   END SELECT
   ENDIF

! 1 INDICATES NORTH POLE, 73 OR 96 IS SOUTH POLE

   END SUBROUTINE LAT_BRACKETS
!                  7
!----6-------------------------------------------------------------------2
   SUBROUTINE ALT_BRACKETS(XALT, I, J, K, L)
!
!  FINDS I-L VALUES FOR SPLINE_4PT FROM ALTITUDE GRID BASED ON
!   XALT
!  step size for data is 1 km, starts at 0 km
!
   REAL :: XALT, XN, upper, lower
   REAL, DIMENSION(96,144,9,91) :: DTE
   INTEGER :: I, J, K, L, N
!
! Look for a match
       nloop: DO N=1,91
    IF (XALT == REAL(N-1)) THEN !ON A GRID POINT
    I=N
    J=N
    K=N
    L=N
    RETURN
    END IF
   END DO nloop
```

```
! else use closest value
  J=INT(XALT+1)
  SELECT CASE (J)
   CASE(1)
   I = 1
   K = 2
   L = 2
   CASE(90)
   I = 90
   K = 91
   L = 91
   CASE DEFAULT
   I=J-1
   K=J+1
   L=J+2
  END SELECT
! 1 = groud, 91 = edge of space

  END SUBROUTINE ALT_BRACKETS
!                 7
!----6-------------------------------------------------------------------2
  SUBROUTINE DATE2YMD(DSTR,Y,M,D)
   CHARACTER(10), INTENT(IN)::DSTR
   INTEGER, INTENT(OUT)::Y,M,D
   INTEGER::CHAR2INT
        CHARACTER(3)::DIAGNOSE='NO!'
! CONVERT DATESTRING TO INTEGERS
   IF (DSTR(8:8).EQ.',') THEN !DATE IS MM/YYYY
        M=CHAR2INT(DSTR(1:2),2)
                   Y=CHAR2INT(DSTR(4:7),4)
                   D=0
   ELSE      !DATE IS YYYY/MM/DD
        M=CHAR2INT(DSTR(6:7),2)
                   Y=CHAR2INT(DSTR(1:4),4)
                   D=CHAR2INT(DSTR(9:10),2)
   ENDIF
   IF (DIAGNOSE.EQ.'YES')WRITE(40,*) 'IN:', DSTR
!   IF (DIAGNOSE.EQ.'YES')WRITE(*,*) 'IN:', DSTR
   IF (DIAGNOSE.EQ.'YES')WRITE(40,*) 'OUT:', Y,M,D
!   IF (DIAGNOSE.EQ.'YES')WRITE(*,*) 'OUT:', Y,M,D
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
  SUBROUTINE FT2KM(F,K)
        REAL, INTENT(IN) :: F
        REAL, INTENT(OUT) :: K
  K=F*0.0003048
  END SUBROUTINE FT2KM
!                 7
!----6-------------------------------------------------------------------2
  SUBROUTINE KM2FT(K,F)
        REAL, INTENT(IN) :: K
        REAL, INTENT(OUT) :: F
  F=K/0.0003048
  END SUBROUTINE KM2FT
!                 7
!----6-------------------------------------------------------------------2
! END OF SUBS AND FUNCTIONS TO GET DOSE RATE
  SUBROUTINE GETINI
        CHARACTER(10)::INIVAR
   CHARACTER(12)::VIEWER,INIVAL
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE,DIAGS
   CHARACTER(1)::JUNK

   REAL::DEFAULTMV

   COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!  WRITE(*,*) 'Reading CARI-PN.INI'
   DIAGS='NO!'
!   DIAGS='YES'
```

```
        OPEN (UNIT=99,FILE='CARI-PN.INI',STATUS='OLD',ACTION='READ')
             DO
   READ(99,9101) INIVAR,JUNK,INIVAL
!
   IF (DIAGS.EQ.'YES') WRITE(40,*) INIVAR, JUNK, INIVAL
!   ENDIF
      IF (INIVAR(1:2).EQ.'VI') THEN
    VIEWER=INIVAL
                  IF (DIAGS.EQ.'YES') WRITE(40,*) INIVAR, VIEWER
   ENDIF
      IF (INIVAR(1:2).EQ.'OU') THEN
    OUTPUT=INIVAL(1:4)
                  IF (DIAGS.EQ.'YES') WRITE(40,*) INIVAR, OUTPUT
   ENDIF
      IF (INIVAR(1:2).EQ.'OS') THEN
    OS=INIVAL(1:5)
                  IF (DIAGS.EQ.'YES') WRITE(40,*) INIVAR, OS
   ENDIF
      IF (INIVAR(1:2).EQ.'DI') THEN
      IF (INIVAR(1:3).EQ.'DIS') THEN
     DISPLAY=INIVAL(1:3)
                   IF (DIAGS.EQ.'YES') WRITE(40,*) INIVAR, DISPLAY
    ELSE
     DIAGNOSE=INIVAL(1:3)
                   IF (DIAGS.EQ.'YES') WRITE(40,*) INIVAR, DIAGNOSE
    ENDIF
           ENDIF
      IF (INIVAR(1:2).EQ.'ME') THEN
    MENUS = INIVAL(1:3)
                  IF (DIAGS.EQ.'YES') WRITE(40,*) INIVAR, MENUS
   ENDIF
            IF (INIVAR(1:2).EQ.'EN') EXIT
   ENDDO
9101  FORMAT(A10,A2,A12)
   CLOSE(99)
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
! END OF SUBS AND FUNCTIONS TO GET DOSE RATE
!
!10000-19999
! BEGIN SUBS AND FUNCTIONS TO MANIPULATE/EVALUATE FLIGHT PROFILE DATA
! A. BIG FILES (SEVERAL FLIGHTS, ASSUMES GREAT CIRCLE ROUTES, AIRPORTS)
!  1. CREATE NEW BIG FILE
!  2. CREATE NEW SHORT LIST FOR AN EXISTING BIG FILE
!  3. DELETE EXISTING BIG FILE
!  4. DELETE EXISTING SHORT LIST
!  5. ADD FLIGHTS TO EXISTING BIG FILE
!  6. REVIEW FLIGHTS IN AN EXISTING BIG FILE
!  7. REVIEW FLIGHTS IN AN EXISTING SHORT LIST
!  8. REMOVE FLIGHTS FROM AN EXISTING BIG FILE
! B. DEG FILES (WAYPOINT FILES FOR SINGLE FLIGHTS)
!  1. CREATE A NEW DEG FILE
!  2. DELETE AN EXISTING DEG FILE
!  3. CREATE/EDIT/DELETE A FLIGHT LIST OF DEG FILES
!  A. ADD ALL DEG FILES IN CURRENT DIRECTORY TO LIST
!  B. ADD ONLY SELECTED DEG FILES
!  C. REMOVE FILES FROM AN EXISTING LIST
! C. EVALUATE FLIGHT PROFILES OUTPUT
!  1. RESULTS TO ARCHIVE
!  2. RESULTS TO SCREEN
!  3. RESULTS TO ARCHIVE AND SCREEN
! D. EVALUATE FLIGHTS
! RDBIGFLIGHT READS THE WHOLE FLIGHT PROFILE FOR ANALYSYS BY SUBROUTINE
! BIG_FLT_DOSE
  SUBROUTINE RDBIGFLT(FLTNAME,YMD,OPORT,DPORT,CLIMBMIN,NOSTEPS,  &
  & STEPFEET,STEPMIN,DESCMIN,CRUISEMIN,TRIPMIN,BAD)
  CHARACTER(30)::FLTNAME,HEADLINE
  CHARACTER(10)::FLTDATE,YMD
  CHARACTER(6)::OPORT,DPORT,ENDSTR
  INTEGER:: NOSTEPS, STEPFEET(12), STEPMIN(12), DESCMIN
```

```
  INTEGER:: CLIMBMIN, CRUISEMIN, TRIPMIN, I, J, BAD
  REAL :: DIST,TRIPMILES,MAXFEET
! Common Block variables from ini file
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT

!  CHARACTER(3)::DIAGNOSE='YES'
  BAD=0
  MAXFEET = 0
        CLIMBMIN = 0
        CRUISEMIN = 0
  DESCMIN = 0
  TRIPMIN = 0
  NOSTEPS = 0
  DO I = 1,12
   STEPFEET(I)=0
   STEPMIN(I)=0
  ENDDO
!  IF (MENUS.EQ.'NO!') THEN
!   READ(18,*) HEADLINE
!  ENDIF
  READ(18,78801,ERR=78810,END=78820) FLTNAME
  READ(18,78802,ERR=78810,END=78820) FLTDATE
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'READING', FLTNAME
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'READING DATE', FLTDATE
  FLTDATE=ADJUSTL(FLTDATE)
  IF ((LEN(TRIM(FLTDATE))).EQ.7) THEN
         IF(FLTDATE(3:3).EQ.'/') THEN ! US MM/YYYY FORMAT
               YMD(1:4)=FLTDATE(4:7)
   YMD(5:5)='/'
   YMD(8:8)='/'
   YMD(6:7)=FLTDATE(1:2)
   YMD(9:10)='00'
   ELSEIF (FLTDATE(5:5).EQ.'/') THEN !INTL / EUROPEAN FORMAT
               YMD(1:4)=FLTDATE(4:7)
   YMD(5:5)='/'
   YMD(8:8)='/'
   YMD(6:7)=FLTDATE(1:2)
   YMD(9:10)='00'
   ELSE
   BAD = 1
   ENDIF
  ELSEIF ((LEN(TRIM(FLTDATE))).EQ.10) THEN
         IF (FLTDATE(5:5).EQ.'/') THEN !EXPECTED LONG FORMAT
               YMD(1:10)=FLTDATE(1:10)
   ELSE
   BAD = 1
   ENDIF
  ELSE
   BAD = 1
  ENDIF
  IF (BAD.EQ.1) THEN
   DO
   READ(18,*,ERR=78810,END=78820) ENDSTR
   IF (SCAN(ENDSTR,'-').NE.0 .OR. SCAN(ENDSTR,'_').NE.0) EXIT
   ENDDO
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'EXITING RDBIGFLT, BAD=',BAD
   RETURN
  ENDIF
  READ(18,*) OPORT
  OPORT = ADJUSTL(OPORT)
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'OPORT', OPORT
  READ(18,*) DPORT
  DPORT = ADJUSTL(DPORT)
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'DPORT', DPORT
  READ(18,*) NOSTEPS
  READ(18,*) CLIMBMIN
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'STEPS', NOSTEPS
```

```
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'CLIMB MIN', CLIMBMIN
  DO I = 1, NOSTEPS
       READ(18,*,ERR=78810,END=78820) STEPFEET(I), STEPMIN(I)
             IF(STEPFEET(I) > MAXFEET ) MAXFEET = STEPFEET(I)
                   ! store highest altitude
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) STEPFEET(I), STEPMIN(I)
  END DO
       READ(18,*,ERR=78810,END=78820) DESCMIN
  READ(18,*,ERR=78810,END=78820) ENDSTR
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) DESCMIN, ENDSTR

  IF(MAXFEET > 60000 ) THEN
       BAD=4
       RETURN
  ENDIF
  DO I = 1, NOSTEPS
   CRUISEMIN = CRUISEMIN + STEPMIN(I)
  END DO
!   *** TripTime is ground to ground time in minutes *****
  TRIPMIN = CRUISEMIN + DESCMIN + CLIMBMIN
78801 FORMAT(A30)
78802 FORMAT(A10)
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'EXITING RDBIGFLT, BAD=',BAD
  RETURN
78810 BAD=2
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'EXITING RDBIGFLT, BAD=',BAD
  RETURN
78820 BAD=3
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'EXITING RDBIGFLT, BAD=',BAD
  END SUBROUTINE
!--------1---------2---------3---------4---------5---------6---------7--
  FUNCTION BADSTR(I)
  CHARACTER(45)::BADSTR
  INTEGER::I
  SELECT CASE (I)
        CASE(1)
             BADSTR=' CANNOT READ DATE...SKIPPING THIS FLIGHT '
        CASE(2)
             BADSTR=' CANNOT READ ALTS...SKIPPING THIS FLIGHT '
        CASE(3)
             BADSTR=' REACHED END OF BIG FILE     '
   CASE(4)
             BADSTR=' ALT > 60,000 FT...SKIPPING THIS FLIGHT  '
        CASE(5)
             BADSTR=' TRIP TIME >1996 MIN...SKIPPING THIS FLIGHT '
   CASE DEFAULT
             BADSTR=' BAD PROFILE DATA...SKIPPING THIS FLIGHT '
  END SELECT
  END FUNCTION
!--------1---------2---------3---------4---------5---------6---------7--
!+---------------- use Geodesic survey method for trip distance ------+
  SUBROUTINE USE_INVERSE (la1, lo1, la2, lo2, miles, FAZ)

        REAL::METERS2MILES,meters
             REAL, INTENT(IN)::la1,lo1,la2,lo2
             DOUBLE PRECISION:: FAZ,dlat1,dlat2,dlon1,dlon2,BAZ,metres

!   CHARACTER(3)::DIAGNOSE='YES'
   CHARACTER(3)::DIAGNOSE='no!'

   dlat1=DBLE(la1)
   dlat2=DBLE(la2)
   dlon1=DBLE(lo1)
   dlon2=DBLE(lo2)
   call inverse(dlat1,dlon1,dlat2,dlon2,FAZ,BAZ,metres)
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)'FROM LAT= ',la1,' LON=',lo1
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)'TO LAT= ',la2,' LON=',lo2
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)'ALONG FAZ= ',FAZ, ' IS M=', &
  & metres, ' METERS'

   meters=real(metres,kind=4)
   METERS2MILES = 1. / 1852.
```

```
        miles = meters * METERS2MILES

   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) ' ',miles,' NAUTICAL MILES'
  END SUBROUTINE USE_INVERSE
!--------1---------2---------3---------4---------5---------6---------7--
!+--use Geodesic survey method for intermediate coordinates--+
! for best accurracy, miles should be less than 100, BUT
! for CARI a few millimeters is not an issue, results are
! reversible at all reasonable distances
  SUBROUTINE USE_FORWARD (la1, lo1, la2, lo2, miles, FAZ)

!  LATITUDE IS NORTH=POSITIVE, SOUTH=NEGATIVE
!  LONGITUDE IS EAST POSITIVE, WEST=NEGATIVE

        REAL, INTENT(IN) :: miles, la1, lo1
              REAL, INTENT(OUT) :: la2, lo2
   REAL :: meters
              REAL(8), INTENT(IN) :: FAZ
   DOUBLE PRECISION::dlat1,dlon1,dlat2,dlon2,metres

   CHARACTER(3)::DIAGNOSE='NO!'
!   CHARACTER(3)::DIAGNOSE='YES'
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'FORWARD MILES=',miles,  &
  &            'ALONG FAZ= ',FAZ, ' FROM '
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'LAT= ',la1,' LON= ',lo1

   meters = 1852.* miles
   metres=DBLE(meters)
        dlat1=DBLE(la1)
         dlon1=DBLE(lo1)
   call forward(dlat1,dlon1,FAZ,metres,dlat2,dlon2)
   lo2=real(dlon2, kind=4)
              IF (lo2<-180.0) lo2=lo2+360.0
   la2=real(dlat2, kind=4)
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'IS LAT= ',la2,' LON= ',lo2
  END SUBROUTINE USE_FORWARD
!----0-----------------------------------------------------------------7
!----6-----------------------------------------------------------------2
  SUBROUTINE BIG_FLT_DOSE(FLTNAME,OPORT,DPORT,YMD2,DOSE,DOSEKIND, &
  & BAD)
! THIS SUB GENERATES A SET OF WAYPOINTS BASED ON THE FLIGHT PROFILE
! THEN CALCULATES A FLIGHT DOSE
   REAL::LAT(2000),LON(2000),DEPTH(2000),DR,DT(2000)
   CHARACTER(30)::DAPORT,OAPORT,DCITY,OCITY
   CHARACTER(10)::YMD
   CHARACTER(6)::OPORT,DPORT,ENDSTR
   CHARACTER(1)::ONS,OEW,DEW,DNS
   CHARACTER(10), INTENT(IN)::YMD2
   INTEGER, INTENT(IN):: DOSEKIND
        INTEGER::Y,M,D,HP
   INTEGER::ALLSTEPS, NUMLOCS
   INTEGER::CLIMBSTEPS, DESCSTEPS
   INTEGER::NOSTEPS, STEPFEET(12), STEPMIN(12), DESCMIN
   INTEGER::CLIMBMIN, CRUISEMIN, TRIPMIN, I, J

   CHARACTER(30), INTENT(OUT)::FLTNAME
   INTEGER, INTENT(OUT)::BAD
   REAL, INTENT(OUT)::DOSE

   REAL :: OALT, DALT, NSM, EWS, SPEED, STEPDIST(12)
   REAL :: DIST,TRIPMILES, RHP,TOTALDOSE,ALT,CAS
   REAL :: OLAT, OLON, DLAT, DLON
   REAL(8)::FAZ
   REAL, DIMENSION(96,144,9,91):: DTE
! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE
   CHARACTER(4):: YYYY
   CHARACTER(1):: lastdigit
   CHARACTER(14):: sd1
```

```
   CHARACTER(5):: sd2
   CHARACTER(9):: evf
   CHARACTER(30):: whichfile

   COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT

! fyi
!
!    TABLE=1, Atmospheric Pressure (kPa)
!    TABLE=2, Atmospheric Depth (g/cm**2)
!    TABLE=3, Atmospheric Density (g/cm**3)
!    TABLE=4, Atmospheric Ionization Rate (cm^-3 s^-1)
!    TABLE=5, Silicon Absorbed Dose Rate (uGy/hr)
!    TABLE=6, Tissue Absorbed Dose Rate (uGy/hr)
!    TABLE=7, Tissue Dose Equivalent Rate (uSv/hr)
!    TABLE=8, Ambient Dose Equivalent Rate (uSv/hr)
!    TABLE=9, Effective Dose Rate (uSv/hr)
!
!    DT=1, Ionization rate
!    DT=2, Si ABSORBDED DOSE RATE
!    DT=3, TISSUE ABSORBED DOSE RATE
!    DT=4, TISSUE DOSE EQUIVALENT RATE
!    DT=5, ICRU AMBIENT DOSE EQUIVALENT RATE, H*(10)
!    DT=6, EFFECTIVE DOSE RATE


   IF (DIAGNOSE.EQ.'YES') THEN
               WRITE(40,*) 'ANALYZING A FLIGHT'
               ENDIF
! INITIALIZE ARRAYS
   DO I=1,2000
   DT(I)=0
   DEPTH(I)=0
   LAT(I)=0
   LON(I)=0
   ENDDO
   DR=0
   DOSE=0
   NUMLOCS=0
   ALLSTEPS=0
   WRITE(40,*) 'CALLING RDBIGFLT '
   CALL RDBIGFLT(FLTNAME,YMD,OPORT,DPORT,CLIMBMIN,NOSTEPS,STEPFEET&
  &     ,STEPMIN,DESCMIN,CRUISEMIN,TRIPMIN,BAD)

   IF (DIAGNOSE=='YES') THEN
               WRITE(40,*) 'RDBIGFLT RETURNS FLIGHTNAME: ',FLTNAME
               WRITE(40,*) 'RDBIGFLT RETURNS ORIGIN PORT: ', OPORT
               WRITE(40,*) 'RDBIGFLT RETURNS DESTINATION: ', DPORT
               WRITE(40,*) 'RDBIGFLT RETURNS CLIMB TIME (MINS): ',CLIMBMIN
               WRITE(40,*) 'RDBIGFLT RETURNS NOSTEPS: ',NOSTEPS
   DO I = 1, NOSTEPS
                WRITE(40,*) 'STEP ',I, STEPMIN(I), STEPFEET(I)
   ENDDO
               WRITE(40,*) 'RDBIGFLT RETURNS CRUISE TIME (MIN): ',CRUISEMIN
               WRITE(40,*) 'RDBIGFLT RETURNS DESCENT TIME (MIN): ',DESCMIN
   WRITE(40,*) 'RDBIGFLT RETURNS FLT TIME (MIN): ' ,TRIPMIN
   WRITE(40,*) ' '
   ENDIF
   CALL PORT_INFO(OPORT,OAPORT,OCITY,ONS,OLAT,OEW,OLON,OALT)
   CALL PORT_INFO(DPORT,DAPORT,DCITY,DNS,DLAT,DEW,DLON,DALT)
   IF (DIAGNOSE=='YES') THEN
   WRITE(40,*) 'ORIGIN INFORMATION'
   WRITE(40,*) 'CITY AND PORTNAMES ', OCITY, OAPORT
   WRITE(40,*) 'READING LAT AND LON ', ONS,OLAT,OEW,OLON
   WRITE(40,*) 'READING ALTITUDE ', OALT
   WRITE(40,*) ' '
   WRITE(40,*) 'DESTINATION INFORMATION'
   WRITE(40,*) 'CITY AND PORTNAMES ', DCITY, DAPORT
   WRITE(40,*) 'READING LAT AND LON ', DNS,DLAT,DEW,DLON
   WRITE(40,*) 'READING ALTITUDE ', DALT
   ENDIF
   IF (ONS.EQ.'N') THEN
```

```
            NSM=1.
   ELSE
            NSM=-1.
   ENDIF
   IF (OEW.EQ.'E') THEN
            EWM=1.
   ELSE
            EWM=-1.
   ENDIF
   OLAT=OLAT*NSM
   OLON=OLON*EWM
   LAT(1)=OLAT
   LON(1)=OLON
   IF (DNS.EQ.'N') THEN
            NSM=1.
   ELSE
            NSM=-1.
   ENDIF
   IF (DEW.EQ.'E') THEN
            EWM=1.
   ELSE
            EWM=-1.
   ENDIF
   DLAT=DLAT*NSM
   DLON=DLON*EWM

   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)'CALLING INVERSE'
   CALL USE_INVERSE(OLAT,OLON,DLAT,DLON,miles,FAZ)
!  SPEED FOR ESTIMATING SEGMENT DISTANCES
   SPEED = miles/(CRUISEMIN+0.5*(DESCMIN+CLIMBMIN))

   DESCDIST = SPEED*DESCMIN
        CLIMBSTEPS = CLIMBMIN+1
        DESCSTEPS = DESCMIN+1
!  CLIMBING STEPS, USE ONE STEP PER MINUTE, CENTERED ON 1/2 STEP
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'CALCULATING CLIMB'
   CALL FT2KM(OALT,DEPTH(1))
   DT(1)=0.5
   CAS=(STEPFEET(1)-OALT)/CLIMBMIN
   DO I = 1,CLIMBMIN !ALT,TIME,LOCS DURING CLIMB
            DT(I+1)=1
         CLIMBDIST = 0.5*SPEED*I
   CALL FT2KM(OALT+CAS*I,DEPTH(I+1))
   CALL USE_FORWARD(LAT(1),LON(1),LAT(I+1),LON(I+1),CLIMBDIST, &
  &      FAZ)
   IF(DIAGNOSE.EQ.'YES')WRITE(40,*)I+1,DEPTH(I+1),CLIMBDIST,FAZ
        ENDDO
             DT(CLIMBMIN+1)=0.5

!  CRUISING STEPS [CLIMBMIN+2,CLIMBMIN+CRUISEMIN+2,]
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'CALCULATING CRUISE COORDS'
   IF (TRIPMIN>1996) THEN
            BAD=5 !CAN'T USE MINUTES AS STEPS, TRIP TOO LONG, >33.25h
   TOTALDOSE= -5.0
   RETURN
   ENDIF
   DO I = CLIMBMIN+2,CLIMBMIN+CRUISEMIN+2 !ALT,TIME,LOCS DURING CRUISES
         DT(I)=1
   CRUISEDIST = CLIMBDIST+SPEED*(I-(CLIMBMIN+2)-0.5) !AVE POSITION FOR EACH MIN
   CALL ALTNOW(CLIMBMIN,NOSTEPS,STEPFEET,STEPMIN,DESCMIN,  &
  &                   CRUISEMIN,I,ALT)
   CALL FT2KM(ALT,DEPTH(I))
   CALL USE_FORWARD(LAT(1),LON(1),LAT(I),LON(I),CRUISEDIST,FAZ)
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) I,DEPTH(I),CRUISEDIST,FAZ
        ENDDO
   J=CLIMBMIN+CRUISEMIN+3
   ALLSTEPS=TRIPMIN+4
!  DESCENDING STEPS, USE ONE STEP PER MINUTE, CENTERED ON 1/2 STEP
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'CALCULATING DESCENT'
   CALL FT2KM(DALT,DEPTH(ALLSTEPS))
   DT(J)=0.5
   CAS=(STEPFEET(NOSTEPS)-DALT)/DESCMIN
```

```
   DO I = 1,DESCMIN !ALT,TIME,LOCS DURING CLIMB
               J=J+1
               DT(J)=1
         CLIMBDIST = CRUISEDIST + 0.5*SPEED*I
   CALL FT2KM(STEPFEET(NOSTEPS)-CAS*I,DEPTH(J))
   CALL USE_FORWARD(LAT(1),LON(1),LAT(J),LON(J),CLIMBDIST,FAZ)
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)J, DEPTH(J),CLIMBDIST,FAZ
   ENDDO
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'DESTINATION:',LAT(J),LON(J)
        DT(J)=0.5
! WHICH DATE TO USE YMD2='0000/00/00' FOR USE PROFILE INFO
   IF (YMD2.NE.'0000/00/00') THEN
   YMD=YMD2
   ENDIF
   CALL DATE2YMD(YMD,Y,M,D)
! OPEN CORRECT DATA FILE DOSE RATES
   sd1='NAIRAS_TABLES\'
   sd2=YMD(1:3)//'0\'
   lastdigit=YMD(4:4)
   evf='YEARLY.FF'
   whichfile=sd1//sd2//evf//lastdigit
   OPEN(21,FILE=whichfile,STATUS='OLD')
   CALL LOADER73(DTE)
! CALCULATE FLIGHT DOSE
              IF (DIAGNOSE=='YES') WRITE(40,*) 'CALCULATING FLIGHT DOSE'
   DO I = 1, ALLSTEPS
   DR=FDR(LAT(I),LON(I),DEPTH(I),DOSEKIND,DTE)
        DOSE=DR*DT(I)/60.+DOSE
   IF (DIAGNOSE=='YES') WRITE(40,*)'STEP, LAT, LON, ALT, RATE' &
  &          //'TIME, CUMDOSE'
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)I, LAT(I), LON(I),  &
  &          DEPTH(I), DR, DT(I), DOSE
   ENDDO
   IF (DIAGNOSE=='YES') WRITE(40,*) DOSE, DOSEKIND
   CLOSE(21)
       END SUBROUTINE
!--------1---------2---------3---------4---------5---------6---------7--

! ALTNOW FINDS ALTITUDE DURING THE FLIGHT AT ANY TIME DURING THE CRUISE
  SUBROUTINE ALTNOW(CLIMBMIN,N,STEPFEET,STEPMIN,DESCMIN,CRUISEMIN,I,&
  &     ALT)
  INTEGER::N
  INTEGER::CLIMBMIN,STEPFEET(N),STEPMIN(N),DESCMIN,CRUISEMIN
  INTEGER::I,J,K,M
  REAL::ALT
        ! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!  CHARACTER(3)::DIAGNOSE='YES'

   K=1
   J=STEPMIN(K)
        DO M=1,CRUISEMIN
               IF(M.LT.J) THEN
                ALT = REAL(STEPFEET(K),KIND=4)
   ELSE
    ALT = REAL(STEPFEET(K),KIND=4)
    K=K+1
    J=J+STEPMIN(K)
   ENDIF
   IF (M+CLIMBMIN+2==I) EXIT
   ENDDO
   IF (DIAGNOSE=='YES') WRITE(40,*) I, ALT
  END SUBROUTINE ALTNOW
!--------1---------2---------3---------4---------5---------6---------7--
!  1. EVALUATE AT BIG FILE
!  USER CAN OVERRIDE INTERNAL DATE OR POTENTIAL & MUST SELECT DOSE OUTPUT
  SUBROUTINE RUNBIG !(FILENAME,YMD1,DOSEKIND)
```

```
!
  IMPLICIT NONE
  INTEGER::PARTICLE,DOSEKIND,HP1,BAD,WP

  REAL :: DOSE

  CHARACTER(1)::C
  CHARACTER(10)::YMD1,PARTSTR
  CHARACTER(30)::OUTFILENAME
  CHARACTER(39)::DOSTR
  CHARACTER(30)::FILENAME,FLIGHTNAME
  CHARACTER(45)::BADSTR
  CHARACTER(6)::OPORT,DPORT
! COMMON BLOCK VARIABLES
       CHARACTER(10)::INIVAR
  CHARACTER(12)::VIEWER,INIVAL
  CHARACTER(5)::OS
  CHARACTER(4)::OUTPUT
  CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE
  CHARACTER(1)::NEWFORMAT
  INTEGER::DTEDIM1

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
  COMMON /NAIRAS/NEWFORMAT,DTEDIM1
! LOCALLY OVERRIDE DIAGNOSE
!   DIAGNOSE='YES'
!   DIAGNOSE='NO!'
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*)'MENUS,OS,DISPLAY,VIEWER,OUTPUT'
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) MENUS,OS,DISPLAY,VIEWER,OUTPUT
  IF (MENUS.EQ.'NO!') THEN
        FILENAME='DEFAULT.BIG'
   OPEN(UNIT=45,FILE='DEFAULT.DAT',STATUS='OLD')
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'Reading Default.dat'
   READ(45,18924) YMD1, HP1, PARTICLE, DOSEKIND
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) YMD1,HP1,PARTICLE,DOSEKIND
   CLOSE(45)
   GOTO 18918
  ELSE
        IF(OS.EQ.'WIN '.AND. DISPLAY.EQ.'DOS') THEN
              CALL SYSTEM('DIR /W *.BIG')
        ELSEIF(OS.EQ.'LINUX'.AND.DISPLAY.EQ.'DOS') THEN
              CALL SYSTEM('ls *.BIG')
   ELSE
              CONTINUE !PLACEHOLDER FOR WIN GUI
   ENDIF
   PRINT *,'  WHICH FILE TO READ?'
       READ*, FILENAME
  ENDIF
       CALL CLS
  CALL MENUHEADER
       !  TYPE, 10 (n,p,gamma,e+,e-,mu+,mu-,pi+,pi-,total)
  PRINT*, ' '
18904 WRITE(*,*)'  SELECT DOSE TYPE'
!  PRINT*, '  <1> ATMOSPHERIC DEPTH'       !2
  IF (NEWFORMAT.EQ.'Y') THEN
  PRINT*, '  <1> IONIZATION RATE in AIR'      !4
  PRINT*, '  <2> ABSORBED DOSE RATE in SILICON'    !5
  PRINT*, '  <3> ABSORBED DOSE RATE in TISSUE'    !6
  PRINT*, '  <4> DOSE EQUIVALENT RATE in TISSUE'    !7
  PRINT*, '  <5> ICRU H*(10) AMBIENT DOSE EQUIVALENT RATE' !8
  PRINT*, '  <6> ICRP PUB 103 EFFECTIVE DOSE'     !9
  READ*,DOSEKIND
  IF (DOSEKIND.LT.1 .OR. DOSEKIND.GT.6) THEN
   CALL CLS
       PRINT*, '  Entry must be 1 to 6 '
       GOTO 18904
  ENDIF
  ELSE
  PRINT*, '  <1> IONIZATION RATE in AIR'      !4
!  PRINT*, '  <2> ABSORBED DOSE RATE in SILICON'    !5
  PRINT*, '  <2> ABSORBED DOSE RATE in TISSUE'    !6
  PRINT*, '  <3> DOSE EQUIVALENT RATE in TISSUE'    !7
```

```
  PRINT*, '  <4> ICRU H*(10) AMBIENT DOSE EQUIVALENT RATE' !8
  PRINT*, '  <5> ICRP PUB 103 EFFECTIVE DOSE'     !9
  READ*,DOSEKIND
  IF (DOSEKIND.LT.1 .OR. DOSEKIND.GT.5) THEN
   CALL CLS
        PRINT*, '  Entry must be 1 to 5 '
        GOTO 18904
  ENDIF
  IF (DOSEKIND.NE.1) DOSEKIND=DOSEKIND+1
  ENDIF
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'DOSEKIND=',DOSEKIND
  IF (DIAGNOSE.EQ.'YES') WRITE(*,*) 'DOSEKIND=',DOSEKIND

18906 WRITE(*,*)'  USE DATES IN FLIGHTS <Y,N> ?'
  CALL READKB(C,1)
  IF (C.EQ.'N' .OR. C.EQ.'n') THEN
18910  WRITE(*,*)
   WRITE(*,*) '  ENTER A DATE FOR THESE FLIGHTS <YYYY/MM/DD>'
   WRITE(*,*) '  (YEARLY AVERAGE WILL BE USED for ALL DATES)'
   READ(*,18921) YMD1
   IF (YMD1(5:5).NE.'/' .OR. YMD1(8:8).NE.'/' ) THEN
               WRITE(*,*) 'WRONG FORMAT: RE-ENTER'
    GOTO 18910
   ENDIF
  ELSEIF (C.EQ.'Y' .OR. C.EQ.'y') THEN
   YMD1='0000/00/00'
  ELSE
   WRITE(*,*)'  Please press Y or N key, then Press <ENTER>'
   GOTO 18906
  ENDIF
18918 CONTINUE
  OPEN (UNIT=18,FILE=FILENAME,STATUS='OLD')
!
!  READ PROFILES FROM BIG FILE 1 AT A TIME, PRINT DOSES TO OUTPUT
!
  WP=SCAN(FILENAME,'.',BACK=.TRUE.)
  OUTFILENAME(WP:WP+4)='.OUT'
  OUTFILENAME(1:WP-1)=FILENAME(1:WP-1)
  OPEN (UNIT=17,FILE=OUTFILENAME,STATUS='UNKNOWN')
  WRITE(17,*)'FLIGHTS FROM '//FILENAME
  DO
  IF(DIAGNOSE.EQ.'YES') WRITE(40,*) 'YMD1, DOSEKIND'
  IF(DIAGNOSE.EQ.'YES') WRITE(40,*) YMD1, DOSEKIND
  CALL BIG_FLT_DOSE(FLIGHTNAME,OPORT,DPORT,YMD1,DOSE,DOSEKIND,BAD)
  IF (BAD.EQ.0) THEN
   WRITE(17,18922) FLIGHTNAME,OPORT,DPORT,DOSE,DOSTR(DOSEKIND)
   WRITE(*,*) 'FINISHED ', FLIGHTNAME
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'FINISHED ', FLIGHTNAME
  ELSE
   WRITE(17,18923) BADSTR(BAD)
   IF (BAD.EQ.3) EXIT !QUIT RUN AT END OF FILE
  ENDIF
  ENDDO
  CLOSE(17)
        CLOSE(18)
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'runbig finished'
  IF (DIAGNOSE.EQ.'YES') WRITE(*,*) 'runbig finished'
18920 FORMAT(I4)
18921 FORMAT(A10)
18922 FORMAT(A30,1X,A6,1X,A6,1X,ES11.4,1X,A42)
18923 FORMAT(A45)
18924 FORMAT(A10,I4)
  END SUBROUTINE RUNBIG
!--------1---------2---------3---------4---------5---------6---------7--

!  2. EVALUATE A SHORT LIST
!  3. EVALUATE A WAYPOINT(DEG) FILE
!  4. EVALUATE A LIST OF DEGFILES
! END SUB AND FUNCTIONS TO MANIPULATE/EVALUATE FLIGHT PROFILE DATA
!
!20000-29999
! BEGIN SUBS AND FUNCTIONS TO CREATE AND EVALUATE FILES OF SINGLE LOCATIONS
```

```
!  1. CREATE NEW FILE OF SINGLE LOCATIONS
!  2. ADD NEW LOCATIONS TO AN EXISTING FILE
!  3. EVALUATE A FILE OF SINGLE LOCATIONS
!  A. RESULTS TO ARCHIVE
!  B. RESULTS TO SCREEN
!  C. RESULTS TO ARCHIVE AND SCREEN
!  4. DELETE A FILE OF SINGLE LOCATIONS
!  5. EVALUATE A SINGLE LOCATION
!  A. RESULTS TO ARCHIVE
!  B. RESULTS TO SCREEN
!  C. RESULTS TO ARCHIVE AND SCREEN
! END DATAIN SUBS AND FUNCTIONS
!20000
!--------1---------2---------3---------4---------5---------6---------7--
  SUBROUTINE ONESPOT
! CALCULATE A DOSE RATE AT A SINGLE LOCATION
!
  CHARACTER(1):: lastdigit
  CHARACTER(14):: sd1
  CHARACTER(5):: sd2
  CHARACTER(9):: evf
  CHARACTER(30):: whichfile
  CHARACTER(4)::Y
  CHARACTER(1)::FG,NS,EW,C
  CHARACTER(2)::M,D,LAD,LAM,LAS,LOM,LOS
       CHARACTER(3)::LOD
  CHARACTER(5)::ALTF
  CHARACTER(10)::YMD
  CHARACTER(12)::LATIN,LONIN
  CHARACTER(10)::PARTSTR
  CHARACTER(32)::DSTR(6)
!  CHARACTER(3)::DIAGNOG='YES'
!  CHARACTER(3)::DIAGNOG='no!'
  INTEGER::YEAR,MONTH,DAY,PARTICLE
       INTEGER::DT
  INTEGER::CHAR2INT
  REAL::LAT,LON,GM,W,N,RM,RS
       REAL::FDR,CHAR2REAL,FT,KM
  REAL::DOSERATE,G,RALT,FEET
  REAL, DIMENSION(96,144,9,91):: DTE
       ! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  CHARACTER(1)::NEWFORMAT
  INTEGER::DTEDIM1, J1, J2
  COMMON /NAIRAS/NEWFORMAT,DTEDIM1
  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT

  YEAR=0;MONTH=0;DAY=0

  WRITE(40,*) 'CALLING ONESPOT'
! STEP 1, USER INPUTS LOCATION, PICKS DOSE RATE OUTPUT
! USE DATE

  WRITE(*,*)'  What year?'
  CALL READKB(Y,4);YEAR=CHAR2INT(Y,4)
  WRITE(*,*)'  What month (ENTER 00 FOR YEARLY AVERAGE)?'
  CALL READKB(M,2);MONTH=CHAR2INT(M,2)
  CALL INT2CHAR(MONTH,M,2)
  WRITE(*,*)'  What day (ENTER 00 FOR MONTHLY AVERAGE)?'
  CALL READKB(D,2);DAY=CHAR2INT(D,2)
  CALL INT2CHAR(DAY,D,2)
20000 CONTINUE
  YMD(1:4)=Y
  YMD(6:7)=M
  YMD(9:10)=D
  sd1='NAIRAS_TABLES\'
  sd2=YMD(1:3)//'0\'
  lastdigit=YMD(4:4)
```

```
evf='YEARLY.FF'
whichfile=sd1//sd2//evf//lastdigit
WRITE(*,*)''
WRITE(*,*)'  LOADING DATA FOR ', Y
OPEN(21,FILE=whichfile,STATUS='OLD')
     CALL LOADER73(DTE)
WRITE(40,*)'  The annual average for ',Y, ' is loaded.'
WRITE(40,*)'  Filename: ', whichfile
WRITE(*,*)''
WRITE(*,*)'  LATITUDE'
WRITE(*,*)'  NORTH OR SOUTH <S/N>?'
CALL READKB(NS,1)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) NS
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) NS
IF (NS.EQ.'N') THEN
       N=1.
ELSE
 N=-1.
ENDIF
WRITE(*,*)'  DEGREES LATITUDE (UP TO 12 CHARACTERS).'
CALL READKB(LATIN,12)
IF (SCAN(LATIN,'.').NE.0) THEN
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'USING FRACTIONAL DEGREES'
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) 'USING FRACTIONAL DEGREES'
LAT=N*CHAR2REAL(LATIN,12)
ELSE
LAD=TRIM(LATIN(1:2))
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LAD
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LAD
WRITE(*,*)'  MINUTES (UP TO 2 CHARACTERS).'
CALL READKB(LAM,2)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LAM
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LAM
WRITE(*,*)'  SECONDS (UP TO 2 CHARACTERS).'
CALL READKB(LAS,2)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LAS
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LAS
RD=CHAR2INT(LAD,2)/1.
RM=CHAR2INT(LAM,2)/60.
RS=CHAR2INT(LAS,2)/3600.
LAT=N*(RD+RM+RS)
ENDIF
WRITE(*,*)'  LONGITUDE'
WRITE(*,*)'  EAST OR WEST <E/W>.'
CALL READKB(EW,1)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) EW
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) EW
IF (EW.EQ.'E') THEN
       W=1.
ELSE
 W=-1.
ENDIF
WRITE(*,*)'  DEGREES (UP TO 12 CHARACTERS).'
CALL READKB(LONIN,12)
IF (SCAN(LONIN,'.').NE.0) THEN
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'USING FRACTIONAL DEGREES'
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) 'USING FRACTIONAL DEGREES'
LON=W*CHAR2REAL(LONIN,12)
ELSE
LOD=TRIM(LONIN(1:3))
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LOD
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LOD
RD=CHAR2INT(LOD,3)/1.
WRITE(*,*)'  MINUTES (UP TO 2 CHARACTERS).'
CALL READKB(LOM,2)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LOM
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LOM
RM=CHAR2INT(LOM,2)/60.
WRITE(*,*)'  SECONDS (UP TO 2 CHARACTERS).'
CALL READKB(LOS,2)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LOS
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LOS
```

```
  RS=CHAR2INT(LOS,2)/3600.
  LON=W*(RD+RM+RS)
  ENDIF
  PRINT*, ' '
20001 WRITE(*,*)'  ALTITUDE'
  WRITE(*,*)'  ARE UNITS FT OR KM <F/K>.'
  CALL READKB(FG,1)
  IF (FG.EQ.'F') THEN
   WRITE(*,*)'  ENTER AN ALTITUDE <0 - 258,000 ft>.'
   READ*, RALT
   IF (RALT.GT.258000.0 .OR. RALT.LT.0.0)THEN
   PRINT *, '  INPUT OUT OF RANGE'
   GOTO 20001
   ENDIF
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)FG, RALT
   IF (DIAGNOSE.EQ.'YES') WRITE(*,*) FG, RALT
   CALL FT2KM(RALT,KM)
  ELSEIF (FG.EQ.'K') THEN
   WRITE(*,*)'  ENTER AN ALTITUDE <0 - 90 km>.'
   READ*, RALT
   IF (RALT.GT.90.0 .OR. RALT.LT.0.0)THEN
   PRINT *, '  INPUT OUT OF RANGE'
   GOTO 20001
   ENDIF
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)FG, RALT
   IF (DIAGNOSE.EQ.'YES') WRITE(*,*) FG, RALT
        KM=RALT
  ELSE
   GOTO 20001
  ENDIF
  PRINT*, ' '
  PRINT*, ' '
20004 WRITE(*,*)'  SELECT DOSE TYPE'
  IF (NEWFORMAT.EQ.Y) THEN
  PRINT*, '  <1> IONIZATION RATE in AIR'       !4
  PRINT*, '  <2> ABSORBED DOSE RATE in SILICON'    !5
  PRINT*, '  <3> ABSORBED DOSE RATE in TISSUE'    !6
  PRINT*, '  <4> DOSE EQUIVALENT RATE in TISSUE'    !7
  PRINT*, '  <5> ICRU H*(10) AMBIENT DOSE EQUIVALENT RATE' !8
  PRINT*, '  <6> ICRP PUB 103 EFFECTIVE DOSE'     !9
  READ*,DT
  IF (DT.LT.1 .OR. DT.GT.6) THEN
   CALL CLS
        PRINT*, '  Entry must be 1 to 6 '
        GOTO 20004
  ENDIF
  ELSE
  PRINT*, '  <1> IONIZATION RATE in AIR'       !4
!  PRINT*, '  <2> ABSORBED DOSE RATE in SILICON'    !5
  PRINT*, '  <2> ABSORBED DOSE RATE in TISSUE'    !6
  PRINT*, '  <3> DOSE EQUIVALENT RATE in TISSUE'    !7
  PRINT*, '  <4> ICRU H*(10) AMBIENT DOSE EQUIVALENT RATE' !8
  PRINT*, '  <5> ICRP PUB 103 EFFECTIVE DOSE'     !9
  READ*,DT
  IF (DT.LT.1 .OR. DT.GT.6) THEN
   CALL CLS
        PRINT*, '  Entry must be 1 to 5 '
        GOTO 20004
  ENDIF
  DT = DT+1
  IF (DT.EQ.1) DT=DT-1
  ENDIF
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) DT
  IF (DIAGNOSE.EQ.'YES') WRITE(*,*) DT
!
! STEP 2, CALCULATE SELECTED DOSE RATE
! YMD IS YYYY/MM/DD
!  IF (DIAGNOSE.EQ.'YES') THEN
   WRITE(40,*) ' SENDING FDR', LAT, LON, KM, DT
!  ENDIF
  DOSERATE = FDR(LAT,LON,KM,DT,DTE)
  CLOSE(21)
```

```
! STEP 3, REPORT DOSE RATE
  DSTR(1) = 'IONIZATION RATE: '
  DSTR(2) = 'Si ABSORBED DOSE RATE: '
  DSTR(3) = 'TISSUE ABS. DOSE RATE: '
  DSTR(4) = 'TISSUE EQ. DOSE RATE: '
  DSTR(5) = 'H*(10) , uSv/h: '
       DSTR(6) = 'EFFECTIVE DOSE RATE: '

  CALL CLS
  WRITE(*,*) 'DATE      ',YMD
  WRITE(*,*) 'LATITUDE     ',LAT, NS
  WRITE(*,*) 'LONGITUDE     ',LON, EW
  WRITE(*,*) 'ALTITUDE     ',KM,' km '
  SELECT CASE (DT)
  CASE(1)
   WRITE(*,*) DSTR(1), DOSERATE, ' /cm^3 s^1 '
  CASE(2:4)
   WRITE(*,*) DSTR(DT), DOSERATE, ' uGy/h'
  CASE DEFAULT
   WRITE(*,*) DSTR(DT), DOSERATE, ' uSv/h'
  END SELECT
  CALL OOPS(' ',1)

  END SUBROUTINE
!                  7
!----6-----------------------------------------------------------------2
! SUB TO CONVERT SMALL INTEGERS TO SHORT CHARACTER STRINGS
! FOR PRINTING DATE FORMATS
!
  SUBROUTINE INT2CHAR(I,CC,L)
        INTEGER, INTENT(IN)::I
        CHARACTER(L), INTENT(OUT)::CC

        IF (I.LT.10 .AND. L.EQ.1) THEN
         IF (I.EQ.0) CC='0'
   IF (I.EQ.1) CC='1'
               IF (I.EQ.2) CC='2'
               IF (I.EQ.3) CC='3'
               IF (I.EQ.4) CC='4'
         IF (I.EQ.5) CC='5'
               IF (I.EQ.6) CC='6'
               IF (I.EQ.7) CC='7'
               IF (I.EQ.8) CC='8'
   IF (I.EQ.9) CC='9'
  ELSEIF (I.LT.10 .AND. L.EQ.2) THEN
         IF (I.EQ.0) CC='00'
   IF (I.EQ.1) CC='01'
               IF (I.EQ.2) CC='02'
               IF (I.EQ.3) CC='03'
               IF (I.EQ.4) CC='04'
         IF (I.EQ.5) CC='05'
               IF (I.EQ.6) CC='06'
               IF (I.EQ.7) CC='07'
               IF (I.EQ.8) CC='08'
   IF (I.EQ.9) CC='09'
  ELSEIF (I.GT.9 .AND. I.LT.100 .AND. L.EQ.2) THEN
   OPEN(UNIT=99,STATUS='SCRATCH')
   WRITE(99,20010) I
   REWIND(99)
   READ(99,20011) CC
   CLOSE(99)
  ELSE
   CALL EPITATH(' NUMBER IN INT2CHAR TOO LARGE ', 29)
  ENDIF
20010 FORMAT(I2)
20011 FORMAT(A2)
  END SUBROUTINE
!                  7
!----6-----------------------------------------------------------------2

21000 SUBROUTINE RUN_LOCATIONS
! RUN ALL LOCATIONS IN A DATABASE
```

```
  WRITE(40,*) 'CALLING READIT'
       CALL READIT
  END SUBROUTINE
!                  7
!----6---------------------------------------------------------------2

  SUBROUTINE READIT
! Return dose rates at a list of locations in PLACES.DAT
  CHARACTER(66)::LOI
  CHARACTER(42)::DRSTR
  CHARACTER(10)::YMD,PARTSTR,ALTNUMCHR
  CHARACTER(6)::CHRIN,ICAO
  CHARACTER(4)::YYYY0,YYYY1
  CHARACTER(1)::EW,NS,FG,C1
  CHARACTER(30)::PORTNAME,CITY
  REAL::LAT,LON,ALTITUDE,DEPTH,PALT
  REAL::CHAR2REAL,NSM,EWM
  REAL::DTE(96,144,9,91)
  INTEGER::Y,M,D,CHAR2INT
  INTEGER::I,HP,RAD,DT,S(9),J
  CHARACTER(1):: lastdigit
  CHARACTER(14):: sd1
  CHARACTER(5):: sd2
  CHARACTER(9):: evf
  CHARACTER(30):: whichfile
       ! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!  CHARACTER(3)::DIAGNOG='YES'
!  CHARACTER(3)::DIAGNOG='no!'
       ! PLACES.DAT IS EQUIVALENT TO OLD DATAIN
  OPEN (UNIT=18, FILE='PLACES.DAT', STATUS='OLD')
  OPEN (UNIT=17, FILE='PLACES.ANS', STATUS='UNKNOWN')
  C1=','
  WRITE(17,21025)
  WRITE(40,*) 'CALLING READIT TO READ LIST OF LOCATIONS'
  IF (DIAGNOG.EQ.'YES') WRITE(*,*) 'READING LIST OF LOCATIONS'
  CHRIN='------' ! INITIALIZE CHRIN
  I=0
  DO
!  READ(18,21100,ERR=21200,END=21200,EOR=21010,ADVANCE='YES')LOI
  READ(18,21100,ERR=21200,END=21200,EOR=21010,ADVANCE='NO')LOI
21010 CHRIN=LOI(1:6)
  I=I+1
  IF (DIAGNOG.EQ.'YES') WRITE(40,*) 'TESTING LINE',I
  IF (DIAGNOG.EQ.'YES') WRITE(40,*) LOI(1:66)
  IF (CHRIN.EQ.'START-') EXIT
  ENDDO

  DO J = 1,9
   S(J)=0
  ENDDO
  YYYY0="1111"
  DO
  I=I+1
!  READ(18,21100,ERR=21200,END=21200,EOR=21020,ADVANCE='YES')LOI
  READ(18,21100,ERR=21200,END=21200,EOR=21020,ADVANCE='NO')LOI
21020 CHRIN=LOI(1:6)
  IF (DIAGNOG.EQ.'YES') WRITE(40,*)CHRIN//LOI(7:66)
  IF (CHRIN.EQ.'STOP--') EXIT
  IF (LOI(1:1).EQ.'C') THEN
   IF (DIAGNOG.EQ.'YES') WRITE(40,*) 'SKIPPING COMMENT LINE'
  ELSEIF ((LOI(1:1).EQ.'N') .OR. (LOI(1:1).EQ.'S')) THEN !FULL DATA
   IF (DIAGNOG.EQ.'YES') WRITE(40,*) 'READING '//CHRIN//LOI(7:66)
   CALL FINDCOMMAS(LOI,66,S)
   NS=LOI(1:1)
   IF (NS.EQ.'S') THEN
   NSM=-1.
```

```
   ELSE
   NSM=1.
   ENDIF
   IF (SCAN(LOI(S(1)+1:S(2)-1),'.').NE.0) THEN
   LAT=CHAR2REAL(LOI(S(1)+1:S(2)-1),S(2)-S(1)-1)
   ELSE
   LAT=REAL(CHAR2INT(LOI(S(1)+1:S(2)-1),S(2)-S(1)-1),KIND=4)
   ENDIF
   EW=LOI(S(3)-1:S(3)-1)
   IF (SCAN(LOI(S(3)+1:S(4)-1),'.').NE.0) THEN
   LON=CHAR2REAL(LOI(S(3)+1:S(4)-1),S(4)-S(3)-1)
   ELSE
   LON=REAL(CHAR2INT(LOI(S(3)+1:S(4)-1),S(4)-S(3)-1),KIND=4)
   ENDIF
   FG=LOI(S(5)-1:S(5)-1)
   IF (SCAN(LOI(S(5)+1:S(6)-1),'.').NE.0) THEN
   ALTITUDE=CHAR2REAL(LOI(S(5)+1:S(6)-1),S(6)-S(5)-1)
   ELSE
   ALTITUDE=REAL(CHAR2INT(LOI(S(5)+1:S(6)-1),S(6)-S(5)-1),  &
  &     KIND=4)
   ENDIF
   IF (EW.EQ.'W') THEN
   EWM=-1.
   ELSE
   EWM=1.
   ENDIF
   YMD= LOI(SCAN(LOI,'/')-4:SCAN(LOI,'/')+5)
   YYYY1=YMD(1:4)
   CALL DATE2YMD(YMD,Y,M,D)
   DT = CHAR2INT(LOI(SCAN(LOI,'D')+1:SCAN(LOI,'D')+1),1)
   WRITE(40,*) 'READING LAT AND LON ', NS,LAT,EW,LON
   WRITE(40,*) 'READING ALTITUDE ', FG, ALTITUDE
   WRITE(40,*) 'READING DATE ', YMD
   WRITE(40,*) 'READING UNITS ', DT
! LOAD DOSE TABLE ONLY AS NEEDED
   IF (YYYY0.NE.YYYY1) THEN
   sd1='NAIRAS_TABLES\'
   sd2=YMD(1:3)//'0\'
   lastdigit=YMD(4:4)
   evf='YEARLY.FF'
   whichfile=sd1//sd2//evf//lastdigit
   OPEN(21,FILE=whichfile,STATUS='OLD')
         CALL LOADER73(DTE)
   ENDIF
   IF (FG.EQ.'F') THEN
               IF (DIAGNOG.EQ.'YES') WRITE (40,*) 'CALLING FT2KM'
    CALL FT2KM(ALTITUDE,DEPTH)
   ELSE
    DEPTH=ALTITUDE
   ENDIF
   WRITE (40,*)'data into FDR is', NSM*LAT,EWM*LON,DEPTH,DT
   DOSERATE=FDR(NSM*LAT,EWM*LON,DEPTH,DT,DTE)
   WRITE(40,21030)NSM*LAT,C1,EWM*LON,C1,REAL(ALTITUDE,KIND=4),C1, &
  &            FG,C1,YMD,C1,DOSERATE,C1,DRSTR(DT)
   WRITE(17,21030)NSM*LAT,C1,EWM*LON,C1,REAL(ALTITUDE,KIND=4),C1, &
  &            FG,C1,YMD,C1,DOSERATE,C1,DRSTR(DT)
! ASSIGN GEOGRAPHIC LOCATION BY CODE
  ELSEIF (LOI(1:1).EQ.'A' .OR. LOI(1:1).EQ.'a') THEN
   DIAGNOG="YES"
   IF (DIAGNOG.EQ.'YES') WRITE(40,*) 'READING '//CHRIN//LOI(7:66)
   CALL FINDCOMMAS(LOI,66,S)
   ICAO= ADJUSTL(TRIM(LOI(S(1)+1:S(2)-1)))
   YMD= LOI(SCAN(LOI,'/')-4:SCAN(LOI,'/')+5)
   CALL DATE2YMD(YMD,Y,M,D)
   DT = CHAR2INT(LOI(SCAN(LOI,'D',BACK=.TRUE.)+1:     &
  &      SCAN(LOI,'D',BACK=.TRUE.)+1),1)
   FG=LOI(S(3)-1:S(3)-1)
   IF (SCAN(LOI(S(3)+1:S(4)-1),'.').NE.0) THEN
   ALTNUMCHR=LOI(S(3)+1:S(4)-1)
   ALTITUDE=CHAR2REAL(ALTNUMCHR,10)
   WRITE(40,*) 'ALTITUDE is ', FG, ALTITUDE
   WRITE(40,*) 'Derived from ', LOI(S(3)+1:S(4)-1)
```

```
   ELSE
   ALTNUMCHR=LOI(S(3)+1:S(4)-1)
   ALTITUDE=REAL(CHAR2INT(ALTNUMCHR,10),KIND=4)
   WRITE(40,*) 'ALTITUDE is ', FG, ALTITUDE
   WRITE(40,*) 'Derived from ', LOI(S(3)+1:S(4)-1)
   ENDIF
   CALL PORT_INFO(ICAO,PORTNAME,CITY,NS,LAT,EW,LON,PALT)
   WRITE(40,*) 'CODE ', ICAO
   WRITE(40,*) 'LAT AND LON ', NS,LAT,EW,LON
   WRITE(40,*) 'ALTITUDE ', FG, ALTITUDE
   WRITE(40,*) 'DATE ', YMD
   WRITE(40,*) 'UNITS ', DT
   IF (NS.EQ.'S') THEN
   NSM=-1.
   ELSE
   NSM=1.
   ENDIF
   IF (EW.EQ.'W') THEN
   EWM=-1.
   ELSE
   EWM=1.
   ENDIF
   IF (FG.EQ.'F') THEN
   IF (DIAGNOG.EQ.'YES') WRITE (40,*) 'CALLING FT2KM'
   CALL FT2KM(ALTITUDE,DEPTH)
   ELSE
   DEPTH=ALTITUDE
   ENDIF
   sd1='NAIRAS_TABLES\'
   sd2=YMD(1:3)//'0\'
   lastdigit=YMD(4:4)
   evf='YEARLY.FF'
   whichfile=sd1//sd2//evf//lastdigit
   OPEN(21,FILE=whichfile,STATUS='OLD')
        CALL LOADER73(DTE)

   DOSERATE=FDR(NSM*LAT,EWM*LON,DEPTH,DT,DTE)
   WRITE(40,*)'DOSE RATE AT ', ICAO,' IS ', DOSERATE, DRSTR(DT)
   WRITE(17,21030)NSM*LAT,C1,EWM*LON,C1,REAL(ALTITUDE,KIND=4),C1, &
  &           FG,C1,YMD,C1,DOSERATE,C1,DRSTR(DT)
   DIAGNOG='NO!'
  ELSE
   WRITE(40,*)'COMMENT OR IMPROPER FORMAT FOR DATA AT LINE: ', I
   WRITE(17,*)'COMMENT OR IMPROPER FORMAT FOR DATA AT LINE: ', I, &
  &    'IN PLACES.DAT'
  ENDIF
  ENDDO
  CLOSE(17)
  RETURN
       ! TABLE HEADER
21025 FORMAT('LAT,  LON,  ALTITUDE, DATE,  HP(MV), ', &
  &  'PARTICLE, DOSE RATE,   QUANTITY ')
       ! TABLE CONTENTS
21030 FORMAT(F10.5,A1,F10.5,A1,F11.4,A1,A1,A1,A10,A1,ES11.4,A1,A45)
21100 FORMAT(A66)
21200 WRITE(40,*) 'CANNOT READ LOCATIONS, FILE IS CORRUPT OR EMPTY'
  CALL OOPS('CANNOT READ LOCATIONS, FILE IS CORRUPT OR EMPTY',49)
  WRITE(17,*) 'CANNOT READ LOCATIONS, FILE IS CORRUPT OR EMPTY'
  CLOSE(17)
  CLOSE(18)
  RETURN
  END SUBROUTINE
!                  7
!----6---------------------------------------------------------------2
  SUBROUTINE FINDCOMMAS(SI,L,COMMAS)
   INTEGER::L,COMMAS(9),RM,LM
   CHARACTER(L)::SI
! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE
```

```
  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!  CHARACTER(3)::DIAGNOSE='YES'

  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'FINDING COMMAS IN'
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) SI
  RM=SCAN(SI,',',BACK=.TRUE.)
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'RM ', RM
  I=0
  LM=1
  DO
   I=I+1
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'SCANNING FOR COMMA', I
   COMMAS(I)= SCAN(SI(LM+1:RM+1),',')
   LM=COMMAS(I)+LM
   COMMAS(I)=LM
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'LM ', LM
   IF (LM.EQ.RM .OR. I.GT.9) EXIT
  ENDDO
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'COMMAS AT'
  DO J=1,I
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) COMMAS(I)
  ENDDO
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------2

22000 SUBROUTINE NEWLOCS
! View or revise the locations database.
  WRITE(40,*) 'CALLING NEWLOCS'
  CALL SHOWPICK('PLACES.DAT',10)
       CALL LOCATIONS
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------2
!30000-39999
! BEGIN SUBS AND FUNCTIONS FOR AIRPORT DATA MANIPULATION
! A. LOOK UP AN EXISTING AIRPORT GIVEN A CITY NAME
! B. LOOK UP AN EXISTING AIRPORT GIVEN AN AIRPORT NAME
! C. LOOK UP AN EXISTING AIRPORT GIVEN AN ICAO CODE
30000 SUBROUTINE PORT_INFO(ICAO_IN,APORT,CITY,NS,LAT,EW,LON,ALT)

   INTEGER::I,N,K,CHAR2INT
   INTEGER::ALTF,LOND,LONM,LONS,LATD,LATM,LATS
             REAL::LON,LAT,ALT
   CHARACTER(6), INTENT(IN)::ICAO_IN
   CHARACTER(6)::ICAO,TESTCODE
         CHARACTER(3)::ALTCODE
   CHARACTER(1)::LONEW,LATNS,NSS,EWS,NS,EW
         CHARACTER(30)::APORT,CITY
   CHARACTER(91)::PORTINFO
! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!        CHARACTER(3)::DIAGNOSE='YES'
   TESTCODE=ADJUSTL(ICAO_IN)
             ! FIND THE RECORD
  OPEN (UNIT=98, FILE='PORT.NDX',STATUS='OLD')
  DO
   READ(98,FMT=30008,ERR=30007,END=30007)PORTINFO
   ICAO=PORTINFO(31:36)
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) TESTCODE, ICAO
   IF (LEN_TRIM(ICAO).EQ.4) THEN
   IF (TESTCODE(1:4).EQ.ICAO(1:4)) THEN
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'MATCHED ',ICAO_IN, ICAO
   EXIT
   ENDIF
   ENDIF
```

```
   IF (LEN_TRIM(ICAO).EQ.5) THEN
   IF (TESTCODE(1:5).EQ.ICAO(1:5)) THEN
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'MATCHED ',ICAO_IN, ICAO
   EXIT
   ENDIF
   ENDIF
   IF (LEN_TRIM(ICAO).EQ.6) THEN
   IF (TESTCODE(1:6).EQ.ICAO(1:6)) THEN
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'MATCHED ',ICAO_IN, ICAO
   EXIT
   ENDIF
   ENDIF
  ENDDO
  APORT=PORTINFO(1:30)
  CITY=PORTINFO(37:66)
  LATNS=PORTINFO(74:74)
  LATD=CHAR2INT(PORTINFO(67:68),2)
  LATM=CHAR2INT(PORTINFO(69:70),2)
  LATS=CHAR2INT(PORTINFO(71:72),2)
  LONEW=PORTINFO(83:83)
  LOND=CHAR2INT(PORTINFO(75:77),3)
  LONM=CHAR2INT(PORTINFO(78:79),2)
  LONS=CHAR2INT(PORTINFO(80:81),2)
  ALTF=CHAR2INT(PORTINFO(84:89),6)
  LON=REAL(LOND,KIND=4)+(LONM/60.)+(LONS/3600.)
  LAT=REAL(LATD,KIND=4)+(LATM/60.)+(LATS/3600.)
  ALT=REAL(ALTF,KIND=4)/10.
  NS=NSS(LATNS)
  EW=EWS(LONEW)
  CLOSE(98)
  IF (DIAGNOSE.EQ.'YES') THEN
   WRITE(40,30001) ICAO_IN
   WRITE(40,30002) APORT
   WRITE(40,30003) CITY
   WRITE(40,30004) LATD, LATM, LATS, NS
   WRITE(40,30005) LOND, LONM, LONS, EW
   WRITE(40,30006) REAL(ALTF,KIND=4)/10.0
       ENDIF
30001 FORMAT(10X,'ICAO CODE: ',A6, '    OTHER CODE: ',A3)
30002 FORMAT(10X,'AIRPORT NAME: ',A30)
30003 FORMAT(10X,'CITY NAME: ',A30)
30004 FORMAT(10X,'LATITUDE: ',I2,' DEGS ',I2,' MINS ',I2,' SECS ',A1)
30005 FORMAT(10X,'LONGITUDE: ',I3,' DEGS ',I2,' MINS ',I2,' SECS ',A1)
30006 FORMAT(10X,'ALTITUDE: ',F7.1, ' FEET')
  RETURN
30007 CLOSE(98)
  CALL OOPS ('ICAO CODE NOT FOUND IN DATABASES',32)
30008 FORMAT(A91)
  END SUBROUTINE
! D. LOOK UP AN EXISTING AIRPORT GIVEN AN IATA CODE
! E. ADD A NEW AIRPORT TO THE DATABASE
! F. UPDATE AIRPORT DATABASE INFO FOR AN EXISTING AIRPORT
! END SUBS AND FUNCTIONS FOR AIRPORT DATA
!
!40000-49999
! BEGIN SUBS TO VIEW HELP FILE
! A. SHOW HELP FILE ONE PAGE AT A TIME
! END HELP FILE VIEWING SUBS AND FUNCTIONS
!
!50000-59999
! BEGIN SUBS AND FUNCTIONS FOR LOOKUP/REVISING OF HELIOCENTRIC POTENTIALS
! A. LOOK UP A HELIOCENTRIC POTENTIAL, GIVEN A DATE
! B. ADD NEW PERMANENT POTENTIALS TO THE DATABASE
! C. TEMPORARILY CHANGE POTENTIALS IN THE DATABASE
! END SUBS AND FUNCTIONS FOR HELIOCENTRIC POTENTIALS
!
!60000-69999
! BEGIN SUBS AND FUNCTIONS FOR GREAT CIRCLE ROUTES
! END SUBS AND FUNCTIONS FOR GREAT CIRCLE ROUTES
!
!70000-99999 MENUS
!                  7
```

```
!----6--------------------------------------------------------------2
  SUBROUTINE MAINMENU
  CHARACTER(1)::CHOICE
70000 CALL CLS          ! BEGIN INPUT LOOP
  CALL MENUHEADER
  WRITE(*,70004)'     MAIN MENU      '
  PRINT*, ' '
  WRITE(*,70004)'<1> HELP file (Read me).       '
  PRINT*, ' '
  WRITE(*,70004)'<2> Galactic radiation received on flights.  '
  PRINT*, ' '
  WRITE(*,70004)'<3> View, add, or change airport information. '
  PRINT*, ' '
  WRITE(*,70004)'<4> Radiation level at user-specified   '
  WRITE(*,70004)'  altitude and geographic coordinates.   '
  PRINT*, ' '
  WRITE(*,70004)'<5> Change output settings. View old results. '
  PRINT*, ' '
  WRITE(*,70004)'<6> Exit program.        '
  PRINT*, ' '
  WRITE(*,70001)'.'
  CALL READKB(CHOICE,1)
  IF (CHOICE.EQ.'q'.OR.CHOICE.EQ.'Q') CHOICE='7'
  IF (CHOICE.EQ.'1'.OR.CHOICE.EQ.'2'.OR.CHOICE.EQ.'3'.OR.   &
  & CHOICE.EQ.'4'.OR.CHOICE.EQ.'5'.OR.CHOICE.EQ.'6') THEN
   GOTO 70002
  ELSE
   GOTO 70000 ! TRY AGAIN, NOT A VALID CHOICE
  END IF
70002 CONTINUE
! DIAGNOSTIC PRINT*, CHOICE
  IF (CHOICE.EQ.'1') CALL SHOWHELP
  IF (CHOICE.EQ.'2') CALL FLIGHTS
  IF (CHOICE.EQ.'3') CALL AIRPORTS
  IF (CHOICE.EQ.'4') CALL LOCATIONS
  IF (CHOICE.EQ.'5') CALL OUTPUTS
  IF (CHOICE.EQ.'6') STOP
  GOTO 70000
70001 FORMAT(10X,'Type 1, 2, 3, 4, 5, 6, or 7 and press <ENTER> ',A1)
70004 FORMAT(10X,A50)
  END SUBROUTINE MAINMENU
!                 7
!----6--------------------------------------------------------------2
! 71000
  SUBROUTINE SHOWHELP
  CHARACTER(8)::STARTHELP
  CALL CLS
  STARTHELP = 'CARI.HLP'
  CALL SHOWPICK(STARTHELP,8)
  END SUBROUTINE SHOWHELP
!                 7
!----6--------------------------------------------------------------2
! 72000
  SUBROUTINE FLIGHTS
       CALL RUNBIG
  END SUBROUTINE
!                 7
!----6--------------------------------------------------------------2
! 73000
  SUBROUTINE AIRPORTS
  CALL AIRPORT_MENU
  END SUBROUTINE
!                 7
!----6--------------------------------------------------------------2
!
  SUBROUTINE AIRPORT_MENU
  CHARACTER(1)::CHOICE
73000 CALL CLS
  CALL MENUHEADER
  WRITE(*,73004)'    AIRPORT MENU      '
  PRINT*, ' '
  WRITE(*,73004)'<1> Find an airport by airport name.    '
```

```
  PRINT*, ' '
  WRITE(*,73004)'<2> Find airport by city.      '
  PRINT*, ' '
  WRITE(*,73004)'<3> Find airport by code.      '
  PRINT*, ' '
  WRITE(*,73004)'<4> Add an airport OR Change airport information.'
  PRINT*, ' '
  WRITE(*,73004)'<5> Associate a non-ICAO code with an airport. '
  PRINT*, ' '
  WRITE(*,73004)'<6> Return to Main Menu.       '
  PRINT*, ' '
  WRITE(*,73004)'<7> Exit program.        '
  PRINT*,''
  PRINT*,''
  PRINT*,''
  WRITE(*,73001) '.'
  CALL READKB(CHOICE,1)
73001 FORMAT(10X,'Type 1, 2, 3, 4, 5, 6, or 7 and press <ENTER> ',A1)
  IF (CHOICE.EQ.'1'.OR.CHOICE.EQ.'2'.OR.CHOICE.EQ.'3'.OR.   &
  & CHOICE.EQ.'4'.OR.CHOICE.EQ.'5'.OR.CHOICE.EQ.'6'.OR.   &
  & CHOICE.EQ.'7') THEN
   GOTO 73002
  ELSE
   GOTO 73000 ! TRY AGAIN, NOT A VALID CHOICE
  END IF
73002 CONTINUE
! DIAGNOSTIC PRINT PRINT*, "SUCCESFUL DATA ENTRY ", CHOICE
  IF (CHOICE.EQ.'1') CALL FINDPORT(1)    !73100
  IF (CHOICE.EQ.'2') CALL FINDPORT(2)    !73100
  IF (CHOICE.EQ.'3') CALL FINDPORT(3)    !73100
  IF (CHOICE.EQ.'4') CALL ADDAPORT     !73400
  IF (CHOICE.EQ.'5') CALL ADDACODE     !73500
  IF (CHOICE.EQ.'7') STOP
  CALL MAINMENU
73003 FORMAT (A1)
73004 FORMAT (10X,A50)
73005 FORMAT (10X,A9)
  END SUBROUTINE
!                  7
!----6-------------------------------------------------------------------2
  SUBROUTINE MAKE_NDX
! THIS SUB CREATES THE PORT.NDX FILE FROM THE AIRPORT DATABASES
! THIS SUB CREATES THE CITY.NDX FILE FROM THE AIRPORT DATABASES
   INTEGER::I,J,K,L,IREC
   INTEGER::MXSIZ
   CHARACTER(91)::PORTINFO(7000),PI2(7000),TESTPORT
   INTEGER::PLATD, PLATM, PLATS, PALTF
   INTEGER::PLOND, PLONM, PLONS
   INTEGER::CHAR2INT
   CHARACTER(1)::PLONEW,PLATNS
   CHARACTER(6)::ICAO
   CHARACTER(6)::ICAO_IN
   CHARACTER(30)::PNAME,CNAME
   LOGICAL::LEXIST
! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!   CHARACTER(3)::DIAGNOSE
!   DIAGNOSE='YES'

!SAMPLE RECORD, 91 CHARACTERS LON
!PORTNAME 1-30
!CODE 31-36
!CITY 37-66
!LAT 67-73 DDMMSSBB
!N/S 74 ANY NON-NUMBER OR } INDICATES S
!LON 75-82
!E/W 83 ANY NON-NUMBER OR } INDICATES E
```

```
!ALT(FT X10)84-89
!CARRAIGE RETURN AND LINEFEED CHARACTERS 90-91
!
  L=91
! INCLUDE USER ADDED AIRPORT LIST
  PRINT*, 'Checking for user entered airports'
  INQUIRE (FILE='NEWPORTS.DAT',EXIST=LEXIST)
  IF (LEXIST) THEN
   J=0
   OPEN(UNIT=32,FILE='AIRPORTS\NEWPORTS.DAT',STATUS='OLD')
  ELSE
   CALL EPITATH('NEWPORTS.DAT IS MISSING ',25)
  ENDIF
  DO
   J=J+1
   READ(32,FMT=30103,ERR=30101)PORTINFO(J)
   IF (DIAGNOSE.EQ.'YES') WRITE(40,30104) J, PORTINFO(J)
   TESTPORT=PORTINFO(J)
   ICAO=TESTPORT(31:36)
   PALTF=CHAR2INT(TESTPORT(84:88),5)
   PNAME=TESTPORT(1:30)
   CNAME=TESTPORT(37:66)
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) ICAO, PNAME, CNAME, PALTF
   IF (TRIM(ICAO).EQ.'----'.AND.PALTF.EQ.99999) THEN
   K=J-1
   EXIT
   ENDIF
  ENDDO
  CLOSE(32)
  GOTO 30102
30101 CALL EPITATH('NEWPORTS.DAT IS CORRUPTED',25)
30102 IREC=K
30103 FORMAT (A91)
30104 FORMAT (I5,2X,A91)
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'LAST GOOD RECORD WAS', IREC
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) ICAO, PNAME, CNAME, PALTF
! INCLUDE PERMANENT AIRPORT LIST
  OPEN(UNIT=27,FILE='AIRPORTS\AIRPORTS.DAT',STATUS='OLD')
  J=0
  PRINT*,'Reading airport data'
  DO
   J=J+1
   READ(27,FMT=30103,ERR=30115)PORTINFO(J+IREC)
   IF (DIAGNOSE.EQ.'YES') WRITE(40,30104) IREC+J, PORTINFO(IREC+J)
   K=IREC+J
   TESTPORT=PORTINFO(K)
   ICAO=TESTPORT(31:36)
   PALTF=CHAR2INT(TESTPORT(84:88),5)
   PNAME=TESTPORT(1:30)
            CNAME=TESTPORT(37:66)
   IF (TRIM(ICAO).EQ.'----'.AND.PALTF.EQ.99999) THEN
   K=K-1
   EXIT
   ENDIF
  ENDDO
30115 CONTINUE
  CLOSE(27)
  DO I = 1,K
   PI2(I)=PORTINFO(I)
  END DO
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'LAST GOOD RECORD WAS', K
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) ICAO, PNAME, CNAME, PALTF
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'SORTING ON AIRPORT NAME'
  CALL SHELLSORT(PI2,L,K+1)
  OPEN (UNIT=98, FILE='PORT.NDX',STATUS='UNKNOWN')
  DO I=1,K
!   WRITE(98,FMT=30103) PORTINFO(I)
   WRITE(98,FMT=30103) PI2(I)
  ENDDO
  PI2(K+1)=PORTINFO(K)
  CLOSE(98)
  IF (DIAGNOSE.EQ.'YES') THEN
```

```
   WRITE(40,*) 'FINISHED WRITING PORT.NDX'
   OPEN (UNIT=98, FILE='PORT.NDX',STATUS='OLD')
   DO J=1,K-100
    READ(98,FMT=30103,ERR=30116)TESTPORT
   IF (MOD(J,100) .EQ. 0) THEN
    WRITE(40,30104) J, TESTPORT
    ICAO=TESTPORT(31:36)
    PNAME=TESTPORT(1:30)
    CNAME=TESTPORT(37:66)
    PLATNS=TESTPORT(74:74)
    PLATD=CHAR2INT(TESTPORT(67:68),2)
    PLATM=CHAR2INT(TESTPORT(69:70),2)
    PLATS=CHAR2INT(TESTPORT(71:72),2)
    PLONEW=TESTPORT(83:83)
    PLOND=CHAR2INT(TESTPORT(75:77),3)
    PLONM=CHAR2INT(TESTPORT(78:79),2)
    PLONS=CHAR2INT(TESTPORT(80:81),2)
    PALTF=CHAR2INT(TESTPORT(84:88),5)
    WRITE(40,*) ICAO, PNAME, CNAME
   ENDIF
   ENDDO
30116 CLOSE(98)
  ENDIF
! NOW SWAP NAME AND CITY, THEN RE-SORT ON CITY NAME
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'swapping city and port'
  DO I=1,K
!   PNAME=PORTINFO(I)(1:30)
!   CNAME=PORTINFO(I)(37:66)
!   PI2(I)=PORTINFO(I)
!       PI2(I)(1:30)=CNAME
!       PI2(I)(37:66)=PNAME
!   PNAME=PORTINFO(I)(1:30)
!   CNAME=PORTINFO(I)(37:66)
        PI2(I)(1:30)=PORTINFO(I)(37:66)
        PI2(I)(37:66)=PORTINFO(I)(1:30)
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) I, PI2(I)
  ENDDO
  CALL SHELLSORT2(PI2,L,K+1) !WORKS
! WRITE PRELIM VERSION OF CITY.NDX
  OPEN (UNIT=98, FILE='CITY.NDX',STATUS='UNKNOWN')
  DO I=1,K
   WRITE(98,FMT=30103) PI2(I)
  ENDDO
  CLOSE(98)
  IF (DIAGNOSE.EQ.'YES') THEN
   WRITE(40,*) 'FINISHED WRITING CITY.NDX'
   OPEN (UNIT=98, FILE='CITY.NDX',STATUS='OLD')
   DO J=1,K-100
   READ(98,FMT=30103,ERR=30118)TESTPORT
   IF (MOD(J,100).EQ. 0) THEN
    WRITE(40,30104) J, TESTPORT
    ICAO=TESTPORT(31:36)
    PNAME=TESTPORT(1:30)
    CNAME=TESTPORT(37:66)
    PLATNS=TESTPORT(74:74)
    PLATD=CHAR2INT(TESTPORT(67:68),2)
    PLATM=CHAR2INT(TESTPORT(69:70),2)
    PLATS=CHAR2INT(TESTPORT(71:72),2)
    PLONEW=TESTPORT(83:83)
    PLOND=CHAR2INT(TESTPORT(75:77),3)
    PLONM=CHAR2INT(TESTPORT(78:79),2)
    PLONS=CHAR2INT(TESTPORT(80:81),2)
    PALTF=CHAR2INT(TESTPORT(84:88),5)
    WRITE(40,*) ICAO, PNAME, CNAME
   ENDIF
             ENDDO
30118 CLOSE(98)
  ENDIF
!  DIAGNOSE='NO!'
  END SUBROUTINE MAKE_NDX
!                 7
!----6---------------------------------------------------------------2
```

```
  SUBROUTINE FINDPORT(I)

   INTEGER::I,J,K,IREC
   INTEGER, DIMENSION(11)::PLATD, PLATM, PLATS, PALTF
   INTEGER, DIMENSION(11)::PLOND, PLONM, PLONS
            INTEGER:: CHAR2INT
   CHARACTER(1), DIMENSION(11)::PLONEW,PLATNS
   CHARACTER(6), DIMENSION(11)::ICAO,SICAO
         CHARACTER(6)::ICAO_IN
   CHARACTER(30), DIMENSION(11)::PNAME,CNAME,SORTED
   CHARACTER(30)::INPUT
   CHARACTER(1)::CHOICE
 ! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!  CHARACTER(3)::DIAGNOSE='YES'

! I=1, SEARCH FOR NEAREST PORTS BY AIRPORT NAME
! I=2, SEARCH FOR NEAREST PORTS BY CITY NAME
! I=3, SEARCH BY ICAO CODE
! USER INPUT TARGET STRING
73112 CONTINUE
  CALL CLS
  CALL MENUHEADER
  SELECT CASE(I)
  CASE(1)
  PRINT*,' '
  PRINT*,'      SEARCH BY AIRPORT   '
  PRINT*,'   Please enter up to 30 characters to match'
  CALL READKB(INPUT,30)
  OPEN (UNIT=98, FILE='PORT.NDX',STATUS='OLD')
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'CALLING FIND11 FOR ',INPUT
  CALL FIND11(PNAME, CNAME, ICAO, INPUT)
  CLOSE(98)
  WRITE(*,73104)
  WRITE(*,73113)
  DO J=1,11
   WRITE(*,73106) PNAME(J), CNAME(J), ICAO(J)
  ENDDO
  CASE(2)
  PRINT*,'      SEARCH BY CITY    '
  PRINT*,'   Please enter up to 30 characters to match'
  CALL READKB(INPUT,30)
  OPEN (UNIT=98, FILE='CITY.NDX',STATUS='OLD')
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'CALLING FIND11 FOR ',INPUT
  CALL FIND11(PNAME, CNAME, ICAO, INPUT)
  CLOSE(98)
  WRITE(*,73103)
  WRITE(*,73113)
  DO J=1,11
   WRITE(*,73106) PNAME(J), CNAME(J), ICAO(J)
  ENDDO
  CASE(3)
  PRINT*,'   Please enter an ICAO code '
  CALL READKB(INPUT,6)
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'CALLING REPORT BY ICAO'
  CALL REP_BY_ICAO(INPUT)
  END SELECT
  PRINT*, ' '
  WRITE(*,73114)'<1> Enter an ICAO code to view a full report. '
  PRINT*, ' '
  WRITE(*,73114)'<2> Try another ICAO code.      '
  PRINT*, ' '
  WRITE(*,73114)'<3> Return to Airport Menu.      '
  PRINT*, ' '
  WRITE(*,73114)'<4> Return to Main Menu.       '
  PRINT*, ' '
  WRITE(*,73114)'<5> Exit program.        '
```

```
  PRINT*,''
  WRITE(*,73111) `.'
73110 CALL READKB(CHOICE,1)
73111 FORMAT(`   Type 1, 2, 3, 4, or 5 and press <ENTER> `,A1)
  IF (CHOICE.EQ.'1'.OR.CHOICE.EQ.'2'.OR.CHOICE.EQ.'3'.OR.   &
  & CHOICE.EQ.'4' .OR.CHOICE.EQ.'5') THEN
   GOTO 73122
  ELSE
   GOTO 73110 ! TRY AGAIN, NOT A VALID CHOICE
  END IF
73122 CONTINUE
  SELECT CASE (CHAR2INT(CHOICE,1))
  CASE(1)
   CALL CLS
   CALL MENUHEADER
   PRINT*,' `
   PRINT*,'   Enter an ICAO code to view a full report. `
   CALL READKB(ICAO_IN,6)
   CALL REP_BY_ICAO(ICAO_IN)
  CASE(2)
   GOTO 73112
  CASE(3)
   CALL AIRPORT_MENU
  CASE(4)
   CALL MAINMENU
  CASE(5)
   STOP
  END SELECT
73103 FORMAT(5X,'CITY',13X,12X,'AIRPORT',21X,'CODE')
73104 FORMAT(5X,'AIRPORT',12X,13X,'CITY',21X,'CODE')
73105 FORMAT(`CODE',13X,'CITY',13X,12X,'AIRPORT')
73106 FORMAT(A30,1X,A30,1X,A6)
73107 FORMAT(A6,1X,A30,1X,A30)
73113 FORMAT(`----------------------------------------------------' &
  &'---------------')
73114 FORMAT(10X,A50)
  END SUBROUTINE
!                   7
!----6---------------------------------------------------------------------2
! 73200
! Fortran 95 shell sort routine
! sorts real numbers into ascending numerical order
! Adapted from a shell sort sub by Andrew Duey, written 5-7-04
! and published for free usage at
! http://www.andrewduey.com/cscs252d.htm (ACCESSED 30 MAY 2012)
! QsortCHAR Adapted for character arrays by Kyle Copeland
  SUBROUTINE SHELLSORT(SORTED, D, N) !-----needs numvals and custlist
! This is where we do the shell sort

        LOGICAL, EXTERNAL :: GTQ !funtion to tell which person goes first
        INTEGER, INTENT (IN) :: D
        INTEGER, INTENT(IN) :: N
!Grab the number of values from the calling code
  INTEGER :: i = 0
        INTEGER :: j = 0
        INTEGER :: increment = 3
!This is the increment which can be adjusted up or down depending
! on condition and size of dataset
         CHARACTER(D):: TEMP_VAL
         CHARACTER(D), INTENT(INOUT), DIMENSION(N) :: SORTED
!Define the customer list to handle whatever size is sent
! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!             CHARACTER(3)::DIAGNOSE='YES'

        IF (DIAGNOSE.EQ.'YES') WRITE (*,*) `Now doing Shell sort'
        IF (DIAGNOSE.EQ.'YES') WRITE (40,*) `Now doing Shell sort'
```

```
        IF (DIAGNOSE.EQ.'YES') WRITE (40,*) i,N-1,increment
        DO
        IF (increment==0) EXIT !check to make sure it's not time to end
              DO i = 0, N - 1
      !this loop increments i which is our starting point for the comparison
                    j=i
                    TEMP_VAL = SORTED(i)
                    DO
     !here in the inner loop is where the comparisons happen
  IF ((j<increment) .OR. (GTQ(TEMP_VAL,SORTED(j-increment),D))) EXIT
      !this loop increments j which is the ending point for the comparison
                          SORTED(j) = SORTED(j - increment)
                          j=j-increment
                    ENDDO
                    SORTED(j)=TEMP_VAL
              ENDDO
              IF ((increment/2) /= 0) THEN
    !make adjustments up and down to the increment
                    increment = increment/2
              ELSE IF     (increment==1) THEN
                    increment = 0
              ELSE
                    increment=1;
              END IF
        ENDDO

        IF (DIAGNOSE.EQ.'YES') WRITE (*,*) 'End of Shell sort'
        IF (DIAGNOSE.EQ.'YES') WRITE (40,*) 'End of Shell sort'
  END SUBROUTINE SHELLSORT
! 73200
!                 7
!----6-------------------------------------------------------------------2
! copy of SHELLSORT for CITY.NDX
  SUBROUTINE SHELLSORT2(SORTED, D, N) !-----needs numvals and custlist
!This is where we do the shell sort

  LOGICAL, EXTERNAL :: GTQ !funtion to tell which person goes first
  INTEGER, INTENT (IN) :: D
  INTEGER, INTENT(IN) :: N !grab the number of values from the calling code
  INTEGER :: i = 0
  INTEGER :: j = 0
  INTEGER :: increment = 3
!This is the increment which can be adjusted up or down
! depending on condition and size of dataset
        CHARACTER(D):: TEMP_VAL
        CHARACTER(D), INTENT(INOUT), DIMENSION(N) :: SORTED
!Define the customer list to handle whatever size is sent
! COMMON block variables
  CHARACTER(12)::VIEWER
  CHARACTER(5)::OS
  CHARACTER(4)::OUTPUT
  CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!             CHARACTER(3)::DIAGNOSE='YES'

        IF (DIAGNOSE.EQ.'YES') WRITE (*,*) 'Now doing 2nd Shell sort'
        IF (DIAGNOSE.EQ.'YES') WRITE (40,*)'Now doing 2nd Shell sort'
        IF (DIAGNOSE.EQ.'YES') WRITE (40,*) i,N-1,increment
        DO
        IF (increment==0) EXIT
!check to make sure it's not time to end
              DO i = 0, N - 1
!this loop increments i which is our starting point for the comparison
                    j=i
                    TEMP_VAL = SORTED(i)
                    DO
!here in the inner loop is where the comparisons happen
  IF ((j<incr2ement) .OR. (GTQ(TEMP_VAL,SORTED(j-increment),D))) EXIT
!this loop increments j which is the ending point for the comparison
                          SORTED(j) = SORTED(j - increment)
                          j=j-increment
```

```
                    ENDDO
                    SORTED(j)=TEMP_VAL
              ENDDO
              IF ((increment/2) /= 0) THEN
!make adjustments up and down to the increment
                    increment = increment/2
              ELSE IF     (increment==1) THEN
                    increment = 0
              ELSE
                    increment=1;
              END IF
        ENDDO

        IF (DIAGNOSE.EQ.'YES') WRITE (*,*) 'End of 2nd Shell sort'
        IF (DIAGNOSE.EQ.'YES') WRITE (40,*) 'End of 2nd Shell sort'
  END SUBROUTINE SHELLSORT2
!                 7
!----6-----------------------------------------------------------------2
  LOGICAL FUNCTION GTQ (a, b, c)
! Greater Than Query
! This function takes airport names or codes as the arguments
! and figures out which sorts out first.

       IMPLICIT NONE
   INTEGER :: c
   CHARACTER(c), INTENT(IN)::a,b
!grab the arguments and format them to make Fortran happy
        GTQ = .FALSE.
!if no other conditions are met then the 2nd AIRPORT comes first
   IF ((a==b) .OR. (LLT ( b , a))) THEN
              GTQ = .TRUE.
        END IF
  END FUNCTION GTQ
!                 7
!----6-----------------------------------------------------------------2
! 73300
  SUBROUTINE REP_BY_ICAO(ICAO_IN)

   INTEGER::I,N,K,CHAR2INT
   INTEGER::ALTF,LOND,LONM,LONS,LATD,LATM,LATS
   CHARACTER(6)::ICAO_IN,ICAO
   CHARACTER(3)::ALTCODE
   CHARACTER(1)::LONEW,LATNS,NSS,EWS
   CHARACTER(30)::APORT,CITY
   CHARACTER(91)::PORTINFO
! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

   COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!  CHARACTER(3)::DIAGNOSE='YES'
! FIND THE RECORD
  OPEN (UNIT=98, FILE='PORT.NDX',STATUS='OLD')
  I=0
  DO
       I=I+1
   READ(98,FMT=73308,ERR=73307,END=73307)PORTINFO
   ICAO=PORTINFO(31:36)
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*) ICAO_IN, ICAO
   IF (LEN_TRIM(ICAO).EQ.4) THEN
   IF (ICAO_IN(1:4).EQ.ICAO(1:4)) EXIT
   ENDIF
   IF (LEN_TRIM(ICAO).EQ.5) THEN
   IF (ICAO_IN(1:5).EQ.ICAO(1:5)) EXIT
   ENDIF
   IF (LEN_TRIM(ICAO).EQ.6) THEN
   IF (ICAO_IN(1:6).EQ.ICAO(1:6)) EXIT
   ENDIF
  ENDDO
  APORT=PORTINFO(1:30)
```

```
  CITY=PORTINFO(37:66)
  LATNS=PORTINFO(74:74)
  LATD=CHAR2INT(PORTINFO(67:68),2)
  LATM=CHAR2INT(PORTINFO(69:70),2)
  LATS=CHAR2INT(PORTINFO(71:72),2)
  LONEW=PORTINFO(83:83)
  LOND=CHAR2INT(PORTINFO(75:77),3)
  LONM=CHAR2INT(PORTINFO(78:79),2)
  LONS=CHAR2INT(PORTINFO(80:81),2)
  ALTF=CHAR2INT(PORTINFO(84:89),6)
  CLOSE(98)
  CALL CLS
  CALL MENUHEADER
  WRITE(*,73301) ICAO_IN, ALTCODE(ICAO_IN)
  WRITE(*,73302) APORT
  WRITE(*,73303) CITY
  WRITE(*,73304) LATD, LATM, LATS, NSS(LATNS)
  WRITE(*,73305) LOND, LONM, LONS, EWS(LONEW)
  WRITE(*,73306) REAL(ALTF,KIND=4)/10.0
73301 FORMAT(10X,'ICAO CODE: ',A6, '    OTHER CODE: ',A3)
73302 FORMAT(10X,'AIRPORT NAME: ',A30)
73303 FORMAT(10X,'CITY NAME: ',A30)
73304 FORMAT(10X,'LATITUDE: ',I2,' DEGS ',I2,' MINS ',I2,' SECS ',A1)
73305 FORMAT(10X,'LONGITUDE: ',I3,' DEGS ',I2,' MINS ',I2,' SECS ',A1)
73306 FORMAT(10X,'ALTITUDE: ',F7.1, ' FEET')
  WRITE(*,*)' '
  CALL OOPS (' ',1)
  CALL AIRPORTS
  RETURN
73307 CLOSE(98)
  CALL OOPS ('ICAO CODE NOT FOUND IN DATABASES',32)
73308 FORMAT(A91)
  END SUBROUTINE
!                 7
!----6-----------------------------------------------------------------2
  FUNCTION NSS(A)
! INTERPRETS NORTH/SOUTH CODING IN AIRPORT DATABASES
  CHARACTER(1)::A,B,NSS
  IF (A.EQ.'1'.OR.A.EQ.'2'.OR.A.EQ.'3'.OR.A.EQ.'4'.OR.A.EQ.'5'.OR. &
  & A.EQ.'6'.OR.A.EQ.'7'.OR.A.EQ.'8'.OR.A.EQ.'9'.OR.A.EQ.'0') THEN
  B='N'
  ELSE
  B='S'
  ENDIF
  NSS=B
  END FUNCTION
!                 7
!----6-----------------------------------------------------------------2
  FUNCTION EWS(A)
! INTERPRETS EAST/WEST CODING IN AIRPORT DATABASES
       CHARACTER(1)::A,B,EWS

  IF (A.EQ.'1'.OR.A.EQ.'2'.OR.A.EQ.'3'.OR.A.EQ.'4'.OR.A.EQ.'5'.OR. &
  & A.EQ.'6'.OR.A.EQ.'7'.OR.A.EQ.'8'.OR.A.EQ.'9'.OR.A.EQ.'0') THEN
  B='W'
  ELSE
  B='E'
  ENDIF
  EWS=B
  END FUNCTION
!                 7
!----6-----------------------------------------------------------------2
  FUNCTION ALTCODE(A)
! FINDS ALTERNATES TO ICAO CODES FROM FILE 'CODES'
   CHARACTER(1) :: JUNK
        CHARACTER(6) :: A,AA
   CHARACTER(4) :: AAA, B
   CHARACTER(3) :: C, ALTCODE
   INTEGER :: I
! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
```

```
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
   AA=ADJUSTL(A); AAA=AA(1:4)
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)'TESTING ALTCODE'
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)'CODE IN:', AAA
   DO
   READ(30,73330,ERR=73331,END=73331) C, JUNK, B
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)'VS:', C, B
   IF(AAA.EQ.B) THEN
    REWIND(30)
    ALTCODE=C
    RETURN
   ENDIF
   ENDDO
73331 CONTINUE
  REWIND(30)
  ALTCODE='---'
73330 FORMAT(A3,A1,A4)
  END FUNCTION
!                  7
!----6---------------------------------------------------------------2
! 73400
  SUBROUTINE FIND11(FIELD1, FIELD2, FIELD3, INPUT)
! SUB TO FIND THE NEAREST 11 RECORDS IN THE SORTED ORDER
        INTEGER::I,J,K,L,N,D
   CHARACTER(91):: TESTPORT(7000)
        CHARACTER(30)::FIELD1(11),TESTNAME(7000)
   CHARACTER(30)::FIELD2(11),INPUT
        CHARACTER(6)::FIELD3(11)
             ! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

   COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!   CHARACTER(3)::DIAGNOSE='YES'
   K=0 !NO MATCH YET
   J=0
            DO
   J=J+1
   READ(98,FMT=73402,ERR=73401)TESTPORT(J)
   TESTNAME(J)=TESTPORT(J)(1:30)
            N=J
   IF (DIAGNOSE.EQ.'YES') WRITE(40,*)'COMPARING <<', TESTNAME(J), &
  & 'AND', INPUT,'>>'
   IF (J.GT.1) THEN
   IF (LLE(TESTNAME(J-1),INPUT) .AND. LGT(TESTNAME(J),INPUT)) THEN
   K=J
   WRITE(40,*)'BEST MATCH IS:'
   WRITE(40,73402) TESTPORT(K)
   EXIT
   ENDIF
   ENDIF
   ENDDO
   N=K
   DO J=K+1,K+5
    READ(98,FMT=73402,ERR=73401,END=73401)TESTPORT(J)
            N=J
   ENDDO
73400   IF (N.LT.11) THEN
    READ(98,FMT=73402,ERR=73401)TESTPORT(J)
        N=N+1
    GOTO 73400
   ENDIF
73401 CONTINUE
73402 FORMAT(A91)

   DO L=1,11
   WRITE(40,73402) TESTPORT(N+1-L)
```

```
   FIELD3(L)=TESTPORT(N+1-L)(31:36)
   FIELD1(L)=TESTPORT(N+1-L)(1:30)
   FIELD2(L)=TESTPORT(N+1-L)(37:66)
   ENDDO

  END SUBROUTINE FIND11
!                 7
!----6---------------------------------------------------------------2
! 73500
  SUBROUTINE ADDAPORT
! Add an airport to MORPORTS.DAT
  CHARACTER(30)::CITY,PORT
  CHARACTER(6)::ICAO,ALTF
  CHARACTER(3)::IATA,LOD
  CHARACTER(2)::LOM,LOS,LAD,LAM,LAS
  CHARACTER(1)::NSS,EWS,YN
  CHARACTER(31)::newports
        CHARACTER(91)::NEWPORT

  INTEGER::CHAR2INT
        ! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT

!  CHARACTER(3)::DIAGNOSE='YES'

  DO I = 1,89
   NEWPORT(I:I)='0'
  ENDDO

  CALL CLS
  CALL MENUHEADER
        PRINT*,' `
  WRITE(*,*)'   AIRPORT ENTRY FORM      `
  PRINT*, ` `
  WRITE(*,*)'   <1> Airport name (UP TO 30 CHARACTERS).'
  CALL READKB(PORT,30)
  IF (DIAGNOSE.EQ.'YES') WRITE(40,73501) PORT
  IF (DIAGNOSE.EQ.'YES') WRITE(*,73501) PORT
  WRITE(*,*)'   <2> City name (UP TO 30 CHARACTERS).'
  CALL READKB(CITY,30)
  IF (DIAGNOSE.EQ.'YES') WRITE(40,73501) CITY
  IF (DIAGNOSE.EQ.'YES') WRITE(*,73501) CITY
  WRITE(*,*)'   <3> ICAO code (UP TO 6 CHARACTERS).'
  CALL READKB(ICAO,6)
  IF (DIAGNOSE.EQ.'YES') WRITE(40,73501) ICAO
  IF (DIAGNOSE.EQ.'YES') WRITE(*,73501) ICAO
73502 WRITE(*,*)'   <4> Add a 3 letter code? (YN)'
  CALL READKB(YN,1)
  IF (YN.EQ.'Y') THEN
   WRITE(*,*)'   Enter a 3-Character code.'
   CLOSE (30)
   CALL SHOWPICK(`CODES',5)
   OPEN (UNIT=30,FILE='AIRPORTS\CODES',STATUS='OLD')
  ELSEIF (YN.EQ.'N') THEN
        CONTINUE
  ELSE
        PRINT*,'   Entry invalid, must be Y or N.'
   PRINT*,''
   GOTO 73502
  ENDIF
  WRITE(*,*)' `
  WRITE(*,*)'   LATITUDE'
  WRITE(*,*)'   NORTH OR SOUTH <S/N>'
  CALL READKB(NSS,1)
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) NSS
  IF (DIAGNOSE.EQ.'YES') WRITE(*,*) NSS
  IF (NSS.EQ.'E') THEN
```

```
      NSS='}'
ELSE
 NSS='0'
ENDIF
WRITE(*,*)'   DEGREES (UP TO 2 CHARACTERS).'
CALL READKB(LAD,2)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LAD
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LAD
WRITE(*,*)'   MINUTES (UP TO 2 CHARACTERS).'
CALL READKB(LAM,2)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LAM
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LAM
WRITE(*,*)'   SECONDS (UP TO 2 CHARACTERS).'
CALL READKB(LAS,2)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LAS
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LAS
WRITE(*,*)' `
WRITE(*,*)'   LONGITUDE'
WRITE(*,*)'   EAST OR WEST <E/W>.'
CALL READKB(EWS,1)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) EWS
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) EWS
IF (EWS.EQ.'E') THEN
      EWS='}'
ELSE
 EWS='0'
ENDIF
WRITE(*,*)'   DEGREES (UP TO 3 CHARACTERS).'
CALL READKB(LOD,3)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LOD
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LOD
WRITE(*,*)'   MINUTES (UP TO 2 CHARACTERS).'
CALL READKB(LOM,2)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LOM
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LOM
WRITE(*,*)'   SECONDS (UP TO 2 CHARACTERS).'
CALL READKB(LOS,2)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) LOS
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) LOS
PRINT*, ' `
WRITE(*,*)'   ALTITUDE IN FEET (UP TO 5 CHARACTERS).'
CALL READKB(ALTF,5)
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) ALTF
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) ALTF
PRINT*, ' `
IF (DIAGNOSE.EQ.'YES') WRITE(40,*) `Defining NEWPORT'
IF (DIAGNOSE.EQ.'YES') WRITE(*,*) `Defining NEWPORT'

NEWPORT(1:30)=PORT
NEWPORT(31:36)=ICAO
NEWPORT(37:66)=CITY
IF(CHAR2INT(LAD,2).GT.9) THEN
 NEWPORT(67:68)=LAD
ELSE
      NEWPORT(67:67)='0'
      NEWPORT(68:68)=LAD
ENDIF
IF(CHAR2INT(LAM,2).GT.9) THEN
 NEWPORT(69:70)=LAM
ELSE
      NEWPORT(69:69)='0'
      NEWPORT(70:70)=LAM
ENDIF
IF(CHAR2INT(LAS,2).GT.9) THEN
 NEWPORT(71:72)=LAS
ELSE
      NEWPORT(71:71)='0'
      NEWPORT(72:72)=LAS
ENDIF
NEWPORT(73:73)='0'
NEWPORT(74:74)=NSS
IF(CHAR2INT(LOD,3).GT.9 .AND. CHAR2INT(LOD,3).LT.100) THEN
```

```
       NEWPORT(75:75)='0'
 NEWPORT(76:77)=LOD
ELSEIF(CHAR2INT(LOD,3).GT.99) THEN
       NEWPORT(75:77)=LOD
ELSE
       NEWPORT(75:76)='00'
 NEWPORT(77:77)=LOD
ENDIF
IF(CHAR2INT(LOM,2).GT.9) THEN
 NEWPORT(78:79)=LOM
ELSE
       NEWPORT(78:78)='0'
       NEWPORT(79:79)=LOM
ENDIF
IF(CHAR2INT(LOS,2).GT.9) THEN
 NEWPORT(80:81)=LOS
ELSE
       NEWPORT(80:80)='0'
       NEWPORT(81:81)=LOS
ENDIF
NEWPORT(82:82)='0'
NEWPORT(83:83)=EWS
IF(CHAR2INT(ALTF,5).GT.9 .AND. CHAR2INT(ALTF,5).LT.100) THEN
       NEWPORT(84:86)='000'
 NEWPORT(87:88)=ALTF
ELSEIF(CHAR2INT(ALTF,5).GT.99 .AND. CHAR2INT(ALTF,5).LT.1000) THEN
       NEWPORT(84:85)='00'
       NEWPORT(86:88)=ALTF
ELSEIF(CHAR2INT(ALTF,5).GT.999 .AND. CHAR2INT(ALTF,5).LT.10000) &
& THEN
       NEWPORT(84:84)='0'
       NEWPORT(85:88)=ALTF
ELSEIF(CHAR2INT(ALTF,5).LT.10) THEN
       NEWPORT(84:87)='0000'
       NEWPORT(88:88)=ALTF
ELSE
 NEWPORT(84:88)=ALTF
ENDIF

NEWPORT(89:89)='0'
NEWPORT(90:90)=CHAR(13)!CR
NEWPORT(91:91)=CHAR(10)!LF

IF (DIAGNOSE.EQ.'YES') THEN
WRITE(40,*) 'NEWPORT DEFINED'
WRITE(*,*) 'CITY NAME: ', NEWPORT(37:66)
WRITE(*,*) 'AIRPORT NAME: ', NEWPORT(1:30)
WRITE(*,*) 'ICAO CODE: ', NEWPORT(31:36),'ALTERNATE CODE: ',IATA
WRITE(40,*) 'CITY NAME: ', NEWPORT(37:66)
WRITE(40,*) 'AIRPORT NAME: ', NEWPORT(1:30)
WRITE(40,*) 'ICAO CODE: ', NEWPORT(31:36),'ALTERNATE CODE: ',IATA
CALL OOPS('_',1)
ENDIF

CALL CLS
      CALL MENUHEADER
WRITE(*,*) 'CITY NAME: ', NEWPORT(37:66)
WRITE(*,*) 'AIRPORT NAME: ', NEWPORT(1:30)
WRITE(*,*) 'ICAO CODE: ', NEWPORT(31:36),'ALTERNATE CODE: ',IATA
WRITE(*,*) 'COORDINATES:'
IF (NSS.NE.'0') THEN
 WRITE(*,*) CHAR2INT(NEWPORT(67:68),2),'DEGS',     &
&    CHAR2INT(NEWPORT(69:70),2),'MINS',    &
&    CHAR2INT(NEWPORT(71:72),2),'SECS SOUTH LAT'
ELSE
 WRITE(*,*) CHAR2INT(NEWPORT(67:68),2),'DEGS',     &
&    CHAR2INT(NEWPORT(69:70),2),'MINS',    &
&    CHAR2INT(NEWPORT(71:72),2),'SECS NORTH LAT'
ENDIF
IF (EWS.NE.'0') THEN
 WRITE(*,*) CHAR2INT(NEWPORT(75:77),3),'DEGS',     &
&    CHAR2INT(NEWPORT(78:79),2),'MINS',    &
```

```
  &    CHAR2INT(NEWPORT(80:81),2),'SECS EAST LAT'
  ELSE
   WRITE(*,*) CHAR2INT(NEWPORT(75:77),3),'DEGS',     &
  &    CHAR2INT(NEWPORT(78:79),2),'MINS',    &
  &    CHAR2INT(NEWPORT(80:81),2),'SECS WEST LAT'
  ENDIF
  WRITE(*,*) 'ALTITUDE: ',CHAR2INT(NEWPORT(84:88),5)
  PRINT*,' '
  PRINT*,'   IS THIS INFORMATION CORRECT <Y/N>?'
  newports='AIRPORTS\NEWPORTS.DAT'
  CALL READKB(YN,1)
       IF (YN.EQ.'Y') THEN
       OPEN(UNIT=32,FILE=newports,STATUS='OLD',POSITION='APPEND')
  BACKSPACE(32)
       WRITE(32,73505) NEWPORT!overwrite 'end of file' entry
  NEWPORT(1:32)='END OF FILE NEWPORTS.DAT  --'
  NEWPORT(33:74)='-- END OF FILE NEWPORTS.DAT  99999990'
  NEWPORT(75:89)='999999999999990'
  WRITE(32,73505) NEWPORT !write new 'end of file' entry
  CLOSE(32)
       CALL MAKE_NDX
  ELSE
  CALL AIRPORT_MENU
  ENDIF
73501 FORMAT ('ENTERED',A30)
73503 FORMAT (A91)
73505 FORMAT (A89)
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
! 73600
  SUBROUTINE ADDACODE
  CALL SHOWPICK('CODES',5)
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
! 74000
  SUBROUTINE LOCATIONS
!  menu for single locations
  CHARACTER(1)::CHOICE
74000 CALL CLS
  CALL MENUHEADER
  WRITE(*,74004)'    LOCATION MENU      '
  PRINT*, ' '
  WRITE(*,74004)'<1> Dose rate at a single location.    '
  PRINT*, ' '
  WRITE(*,74004)'<2> Calculate dose rates for places in the  '
  WRITE(*,74004)'  locations file PLACES.DAT.     '
  PRINT*, ' '
  WRITE(*,74004)'<3> View/Revise the locations file PLACES.DAT '
  WRITE(*,74004)'  using the default text editor.    '
  PRINT*, ' '
  WRITE(*,74004)'<4> View results in PLACES.ANS.     '
  PRINT*, ' '
  WRITE(*,74004)'<5> HELP.          '
  PRINT*, ' '
  WRITE(*,74004)'<6> Return to Main Menu.       '
  PRINT*, ' '
  WRITE(*,74004)'<7> Exit program.        '
  PRINT*,''
  WRITE(*,74001) '.'
  CALL READKB(CHOICE,1)
74001 FORMAT(10X,'Type 1, 2, 3, 4, 5, 6, or 7 and press <ENTER> ',A1)
  IF (CHOICE.EQ.'1'.OR.CHOICE.EQ.'2'.OR.CHOICE.EQ.'3'.OR.   &
  & CHOICE.EQ.'4'.OR.CHOICE.EQ.'5'.OR.CHOICE.EQ.'6'.OR.   &
  & CHOICE.EQ.'7') THEN
   GOTO 74002
  ELSE
   GOTO 74000 ! TRY AGAIN, NOT A VALID CHOICE
  END IF
74002 CONTINUE
! DIAGNOSTIC PRINT PRINT*, "SUCCESFUL DATA ENTRY ", CHOICE
```

```
  IF (CHOICE.EQ.'1') CALL ONESPOT     !20000
  IF (CHOICE.EQ.'2') CALL RUN_LOCATIONS    !21000
  IF (CHOICE.EQ.'3') CALL NEWLOCS     !22000
  IF (CHOICE.EQ.'4') CALL SHOWDOSERATES    !
  IF (CHOICE.EQ.'5') CALL SHOWHELP     !
  IF (CHOICE.EQ.'7') STOP
  CALL MAINMENU
74003 FORMAT (A1)
74004 FORMAT (10X,A50)
74005 FORMAT (10X,A9)
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
! 76000
  SUBROUTINE OUTPUTS
  CHARACTER(1)::CHOICE
  CALL OUTPUT_MENU(CHOICE)
!  DEFAULT IS SET OUTPUT TO FILE ONLY
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
  SUBROUTINE OUTPUT_MENU(CHOICE)
  CHARACTER(1)::CHOICE, OUTPUT, JUNK
  CHARACTER(12)::VARIABLE, SETTING
  CHARACTER(9)::CS(3)
       CHARACTER(50)::MYTEXT
  INTEGER::I
       ! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPT
!  CHARACTER(3)::DIAGNOSE
!       DIAGNOSE='YES'
76000 CALL CLS
  PRINT*, ' '
  CALL MENUHEADER
  WRITE(*,76005)'    OUTPUT MENU      '
  PRINT*, ' '
  WRITE(*,76005)'<1> Edit default settings in CARI.INI.   '
  PRINT*, ' '
  WRITE(*,76005)'<2> Edit default DATE.       '
  PRINT*, ' '
  WRITE(*,76005)'<3> Review existing flight dose archives.  '
  PRINT*, ' '
  WRITE(*,76005)'<4> Review existing location dose rate archives. '
  PRINT*, ' '
  WRITE(*,76005)'<5> Return to Main Menu.       '
  PRINT*, ' '
  WRITE(*,76005)'<6> Exit program.        '
  PRINT*,''
  WRITE(*,76001) '.'
  CALL READKB(CHOICE,1)
76001 FORMAT('  Type 1, 2, 3, 4, 5, or 6 and press <ENTER> ',A1)
  IF (CHOICE.EQ.'1'.OR.CHOICE.EQ.'2'.OR.CHOICE.EQ.'3'.OR.   &
  & CHOICE.EQ.'4'.OR.CHOICE.EQ.'5'.OR.CHOICE.EQ.'6') THEN
   GOTO 76002
  ELSE
   GOTO 76000 ! TRY AGAIN, NOT A VALID CHOICE
  END IF
76002 CONTINUE
! DIAGNOSTIC PRINT
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'SUCCESFUL DATA ENTRY ', CHOICE
  IF (CHOICE.EQ.'1') THEN
   CALL SHOWPICK('CARI-PN.INI',10)
  ELSEIF(CHOICE.EQ.'2') THEN
   CALL SHOWPICK('DEFAULT.DAT',10)
  ELSEIF(CHOICE.EQ.'3') THEN
   CALL SHOWDOSES
  ELSEIF(CHOICE.EQ.'4') THEN
```

```
   CALL SHOWDOSERATES
  ELSEIF(CHOICE.EQ.'6') THEN
   STOP
  ELSE
   CALL MAINMENU
  ENDIF
76003 FORMAT (A1)
76005 FORMAT (10X,A50)
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
  SUBROUTINE SHOWDOSES
! OPEN A FLIGHT DOSE ARCHIVE
  CALL PICKFROMLIST('OUT')
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
  SUBROUTINE SHOWDOSERATES
! OPEN A DOSE RATE ARCHIVE (RESULTS AT SINGLE LOCATIONS)
  CALL SHOWPICK('PLACES.ANS',10)
!  CALL PICKFROMLIST('ANS')
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
  SUBROUTINE PICKFROMLIST(FILE_EXT)
  CHARACTER(2)::SELECTION,GOON
  CHARACTER(3)::FILE_EXT
  CHARACTER(60)::MESSAGE, FILE_LIST(5,16)
  CHARACTER(52)::FILENAME
  INTEGER::FILENUM,SELECTED,PAGE,I,J,M,N,FNUM,CHAR2INT

  CALL SYSTEM('DIR /B *.'//FILE_EXT//' > FILELIST.TXT')
  OPEN(UNIT=60,FILE='FILELIST.TXT',STATUS='OLD')

  N=-1
  DO I=1,5
   DO J = 1,16
   N=N+1
   READ(60,*,ERR=76014,END=76014) FILE_LIST(I,J)
   ENDDO
  ENDDO
76014 CONTINUE
  CLOSE(60)
  SELECT CASE (N)
  CASE (0)  ! No such files to view
   MESSAGE = ' No such files exist in this directory!'
   CALL OOPS(MESSAGE,39)
   RETURN
  CASE (1:16)  ! Show what there is
   MESSAGE = ' '
  CASE (17:80) ! Start with the first page
   MESSAGE = ' '
  CASE DEFAULT ! Start with the first page, but there are more
   MESSAGE = ' ' ! than 80 files to choose from
  END SELECT
  M=16 !number of files to show on a page
  PAGE=1
76009 CALL CLS
   WRITE(*,*) ' '
   PRINT*,' PAGE ',PAGE,' OF ',(N+16)/16
   WRITE(*,*) ' '
   WRITE(*,*)' Showing available files 16 at a time.'
   WRITE(*,*) ' '
! SHOW FILES 1 PAGE AT A TIME
   IF (M>N-(PAGE-1)*16) THEN ! DO NOT HAVE A FULL PAGE
   M=N
   ELSE
   M=16
   ENDIF
   DO I = 1, M !
   WRITE(*,76010) I+(PAGE-1)*16, FILE_LIST(PAGE,I)
   ENDDO
```

```
   PRINT *,' `
   SELECT CASE(PAGE)
               CASE(1)
                IF (N.LT.17) THEN
           WRITE(*,76011)
          ELSE
     WRITE(*,76012)
    ENDIF
         CASE(2:4)
           WRITE(*,76013)
         CASE(5)
           WRITE(*,76015)
         END SELECT
   CALL READKB(SELECTION,2)
  IF (SELECTION=='n' .OR. SELECTION=='N') THEN
   PAGE = PAGE + 1
   GOTO 76009
  ENDIF
  IF (SELECTION=='P' .OR. SELECTION=='p') THEN
   PAGE = PAGE - 1
   GOTO 76009
  ENDIF
  IF (SELECTION=='q' .OR. SELECTION=='Q') RETURN

  FILENUM=CHAR2INT(SELECTION,2)
  FILENAME = FILE_LIST(PAGE,FILENUM)
  CALL SHOWPICK(FILENAME,LEN(FILENAME))
76010 FORMAT (10X,'<',I2,'> `,A60)
76011 FORMAT (10X,'Enter a file number or quit <Q,q>: ` )
76012 FORMAT (10X,'Enter a file number, see next page <N,n>'   &
  & ,', or quit <Q,q>: `)
76013 FORMAT (10X,'Enter a file number, see next page <N,n>'   &
  & ,', see preivous page <P,p>, or quit <Q,q>: `)
76015 FORMAT (10X,'Enter a file number',        &
  & `, see preivous page <P,p>, or quit <Q,q>: `)
  END SUBROUTINE PICKFROMLIST
!                 7
!----6-------------------------------------------------------------------2
  SUBROUTINE SHOWPICK(FILENAME,LOFN)
! OPENS `FILENAME' WITH DEFAULT VIEWER

  CHARACTER(80)::COMMAND
  CHARACTER(LOFN)::FILENAME
  CHARACTER(12)::VIEWER,VARNAME
  CHARACTER(1)::TRASH
  INTEGER::I,LOFN
  I=0
! GET DEFAULT VIEWER FROM FILE
  OPEN (UNIT=99,FILE='CARI-PN.INI',STATUS='OLD')
  DO WHILE (I.EQ.0)
   READ(99,*) VARNAME, TRASH, VIEWER
   IF (VARNAME=='VIEWER') THEN
   I=1
   ELSEIF (VARNAME=='END') THEN
   I=2
   ELSE
   I=0
   ENDIF
  ENDDO
76100 CLOSE(99)
  IF (I==1) THEN
   COMMAND = VIEWER//' `//FILENAME
  ELSE
   COMMAND = `NOTEPAD `//FILENAME
  ENDIF
  CALL SYSTEM(COMMAND)
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
  SUBROUTINE CLS
! MIMICS THE CLEAR SCREEN COMMAND CLS IN QBASIC
!  DO I = 1 , 100 ! 100 FOR LARGE DOS BOXES
```

```
!   PRINT*,' `
!  ENDDO
  CALL SYSTEM('CLS')
  END SUBROUTINE CLS
!                 7
!----6-------------------------------------------------------------2
  SUBROUTINE OOPS(MESSAGE,L)
!
! Equivalent to the obsolete pause statement, with an added message to
! the user
!
  INTEGER::L
  CHARACTER(L)::MESSAGE
  CHARACTER(1)::GOON
        ! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!       CHARACTER(3)::DIAGNOSE='YES'
!  CALL CLS
  WRITE(*,10100) MESSAGE
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) MESSAGE
  PRINT*,' `
  WRITE(*,10101)'Press a character then press <ENTER> to continue'
  READ*, GOON
10100 FORMAT(10X,A60)
10101 FORMAT(10X,A50)
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------2
  SUBROUTINE EPITATH(MESSAGE,L)
!
! Error message to user, then kill program
!
  INTEGER::L
  CHARACTER(L)::MESSAGE
  CHARACTER(1)::GOON
        ! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!  CHARACTER(3)::DIAGNOSE='YES'
!  CALL CLS
  WRITE(*,10100) MESSAGE
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) MESSAGE
  PRINT*,' `
  WRITE(*,10101)'Press a character then press <ENTER> to continue'
  READ*, GOON
  STOP
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*) 'GOODBYE'
10100 FORMAT(10X,A60)
10101 FORMAT(10X,A50)
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------2
  SUBROUTINE MENUHEADER

  WRITE(*,*)' `
  PRINT*,               &
  &' CARI-NAIRAS   FAA - Civil Aerospace Medical Institute'
  PRINT*,               &
  &' SEPT 10, 2014   NASA - Langley Research Center   `
  WRITE(*,*)' `
  WRITE(*,*)' `
  END SUBROUTINE
!                 7
```

```
!----6-------------------------------------------------------------------2
  SUBROUTINE READKB(STROUT,L)
! SPECIAL READ FUNCTION FOR USER INPUT FROM KEYBAORD
! KC, 7 JUNE 2012
  INTEGER::L
  CHARACTER(L)::STRIN,STROUT

  IF (L.GT.1) PRINT*,'(If entering more than a word, enclose answer'&
  &,' in quotes)'
  READ*,STRIN
  CALL LC2UC(STRIN,STROUT,L)
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
  SUBROUTINE LC2UC(STRIN,STROUT,L)
! CONVERTS LOWERCASE LETTERS IN STRINGS TO UPPERCASE
! KC, 7 JUNE 2012
  INTEGER::L,I
  CHARACTER(L)::STRIN,STROUT
  CHARACTER(1)::A
! COMMON block variables
   CHARACTER(12)::VIEWER
   CHARACTER(5)::OS
   CHARACTER(4)::OUTPUT
   CHARACTER(3)::MENUS,DISPLAY,DIAGNOSE

  COMMON /INIT/MENUS,OS,DISPLAY,DIAGNOSE,VIEWER,OUTPUT
!  CHARACTER(3)::DIAGNOSE='YES'
  DO I = 1,L
   A=STRIN(I:I)
   IF (IACHAR(A) .GT. 96 .AND. IACHAR(A).LT. 123) THEN
   ! SHIFT IN ASCII DOWN 32 TO GET UPPERCASE FROM LOWERCASE
   STROUT(I:I)=CHAR(IACHAR(A)-32)
   ELSE
   STROUT(I:I)=STRIN(I:I)
   ENDIF
  ENDDO
  IF (DIAGNOSE.EQ.'YES') WRITE(40,*)'CONVERTED `,STRIN,' TO `,STROUT
  END SUBROUTINE
!                 7
!----6-------------------------------------------------------------------2
  FUNCTION DOSTR(I)
        INTEGER::I
       CHARACTER(39)::DOSTR
!    DK=1, IONIZATION
!    DK=2, Si ABSORBDED DOSE RATE
!    DK=3, TISSUE ABSORBED DOSE RATE
!    DK=4, TISSUE DOSE EQUIVALENT RATE
!    DK=5, ICRU AMBIENT DOSE EQUIVALENT RATE, H*(10)
!    DK=6, EFFECTIVE DOSE RATE
!
   SELECT CASE (I)
    CASE(6)
            DOSTR=' microSv, EFFECTIVE DOSE    `
    CASE(1)
     DOSTR=' per cm^3, IONIZATION IN AIR   `
    CASE(5)
          DOSTR=' microSv, ICRU AMBIENT DOSE EQ. H(*10) `
    CASE(2)
     DOSTR=' microGy, ABSORBED DOSE IN SILICON'
    CASE(3)
     DOSTR=' microGy, ABSORBED DOSE IN TISSUE  `
    CASE(4)
     DOSTR=' microSv, EQUIVALENT DOSE IN TISSUE `
   END SELECT
  END FUNCTION DOSTR
!                 7
!----6-------------------------------------------------------------------2
  FUNCTION DRSTR(I)
        INTEGER::I
       CHARACTER(42)::DRSTR
   SELECT CASE (I)
```

```
    CASE(6)
              DRSTR=' microSv/hr, EFFECTIVE DOSE RATE    `
    CASE(4)
    DRSTR=' microSv/hr, EQUIVALENT DOSE RATE    `
    CASE(5)
          DRSTR=' microSv/hr, AMBIENT DOSE EQ. RATE H(*10) `
    CASE(3)
    DRSTR=' microGy/hr, ABSORBED DOSE RATE IN TISSUE `
    CASE(2)
    DRSTR=' microGy/hr, ABSORBED DOSE RATE IN SILICON'
    CASE(1)
    DRSTR=' /cm^3-s, IONIZATION RATE IN AIR    `
   END SELECT
  END FUNCTION DRSTR
!                  7
!----6-------------------------------------------------------------------2
```

APPENDIX D.
Flight Information for Flights in Table 2

Flight information needed to calculate flight doses using CARI for flights in Table 2 is provided in Table D1. The following simplifications and assumption are built in to CARI-NAIRAS for interpreting the flight profile information:

- Flights follow geodesic routes between airports. This is usually a good assumption, since the geodesic is the shortest route. However, variations from this kind of route are often made for reasons such as weather conditions, congested air traffic, or political boundaries. In most cases, deviations are not large enough to significantly affect the dose calculated for the flight.
- Except for the initial climb and final descent, time spent climbing (or in descent) is counted as time at the following altitude. Since this time is usually short, dose rates climb with altitude, and aircraft usually climb as the flight progresses and fuel load lightens; this almost always leads to a slightly conservative/protective estimate.
- Rates of initial climb and final descent are assumed to be constant. In reality, this will vary by aircraft type and load.

Table D1. Flight information for flights in Table 2.

Origin-Destination	ICAO Codes	Climb and descent times (min)	Cruise altitudes (FL)	Time at altitude (min)
HOUSTON, TX, USA – AUSTIN, TX, USA	KIAH KAUS	11 11	200	5
SEATTLE, WA, USA – PORTLAND, OR, USA	KSEA KPDX	8 11	210	3
MIAMI, FL, USA – TAMPA, FL, USA	KMIA KTPA	13 12	240	10
ST.LOUIS, MO, USA – TULSA, OK, USA	KSTL KTUL	24 17	350	10
TAMPA, FL, USA – ST.LOUIS, MO, USA	KTPA KSTL	27 16	310	76
SAN JUAN, PUERTO RICO – MIAMI, FL, USA	TJSJ KMIA	31 18	310 350	21 64
DENVER, CO, USA – MINNEAPOLIS-ST.PAUL, MN, USA	KDEN KMSP	18 12	330	44
NEW ORLEANS, LA, USA – SAN ANTONIO, TX, USA	KMSY KSAT	26 19	390	25
NEW YORK, NY, USA – SAN JUAN, PUERTO RICO	KJFK TJSJ	29 17	330 370	48 87
LOS ANGELES, CA, USA - HONOLULU, HI, USA	KLAX PHNL	24 17	350	272
CHICAGO, IL, USA – NEW YORK, NY, USA	KORD KJFK	23 17	370	54
HONOLULU, HI, USA – LOS ANGELES, CA, USA	PHNL KLAX	29 29	360 380 400	47 189 14
WASHINGTON, DC, USA – LOS ANGELES, CA, USA	KIAD KLAX	35 16	350	230
TOKYO, JAPAN – LOS ANGELES, CA, USA	RJAA KLAX	27 18	330 370	223 261
MINNEAPOLIS-ST.PAUL, MN, USA – NEW YORK, NY, USA	KMSP KJFK	31 16	370	72
LONDON, UK – DALLAS, TX, USA	EGKK KDFW	19 25	280 310 350 390	16 256 211 57
NEW YORK, NY, USA – CHICAGO, IL, USA	KJFK KORD	24 19	390	66

Table D1. Continued

Origin-Destination	ICAO Codes	Climb and descent times (min)	Cruise altitudes (FL)	Time at altitude (min)
			290	26
DALLAS, TX, USA –	KDFW	19	330	218
LONDON, UK	EGKK	25	350	183
			370	36
LISBON, PORTUGAL –	LPPT	36	350	261
NEW YORK, NY, USA	KJFK	17	390	76
SEATTLE, WA, USA –	KSEA	15	350	172
ANCHORAGE, AK, USA	PANC	19		
CHICAGO, IL, USA –	KORD	14	350	52
SAN FRANCISCO, CA, USA	KSFO	26	390	134
SEATTLE, WA, USA –	KSEA	26	370	203
WASHINGTON, DC, USA	KIAD	18		
NEW YORK, NY, USA –	KJFK	29	350	99
SEATTLE, WA, USA	KSEA	17	390	147
LONDON, UK –	EGLL	36	350	70
NEW YORK, NY, USA	KJFK	18	370	284
SAN FRANCISCO, CA, USA –	KSFO	18	370	110
CHICAGO, IL, USA	KORD	27	410	75
CHICAGO, IL, USA –	KORD	26	330	9
LONDON, UK	EGLL	18	370	383
			330	115
TOKYO, JAPAN –	RJAA	33	350	170
NEW YORK, NY, USA	KJFK	26	370	211
			410	176
			310	49
LONDON, UK –	EGLL	32	350	146
LOS ANGELES, CA, USA	KLAX	19	370	145
			390	241
			350	316
NEW YORK, NY, USA –	KJFK	35	390	192
TOKYO, JAPAN	RJAA	27	410	190
			430	19
LONDON, UK –	EGLL	32	350	175
CHICAGO, IL, USA	KORD	19	390	244
ATHENS, GREECE –	LGAT	25	390	74
NEW YORK, NY, USA	KJFK	20	410	444