NUWC-NPT Technical Report 10,689
8 August 1994

The Use of Two-Station Ionosonde Data for IONCAP/ASAPS Model Validation on Transpolar Two-Hop, High-Frequency Skywave Radio Links

Joseph R. Katan
Submarine Electromagnetic Systems Department

Robert Desourdis
Science Applications International Corporation

Naval Undersea Warfare Center Division
Newport, Rhode Island

PREFACE

This report was prepared under NUWC Detachment New London Job Order No. K50400, *Antarctic Support Program*, principal investigator J. R. Katan (Code 3411). The sponsoring activity was the National Science Foundation, P. Smith (Office of Polar Programs).

The technical reviewer for this report was R. J. Pellowski (Code 3422).

REVIEWED AND APPROVED: 8 August 1994

D. M. Viccione
Head, Submarine Electromagnetic Systems Department

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	8 August 1994	Final

4. TITLE AND SUBTITLE

The Use of Two-Station Ionosonde Data for IONCAP/ASAPS Model Validation on Transpolar Two-Hop, High-Frequency Skywave Radio Links

5. FUNDING NUMBERS

6. AUTHOR(S)

Joseph R. Katan
Robert Desourdis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Undersea Warfare Center
Detachment New London
New London, Connecticut 06320

8. PERFORMING ORGANIZATION REPORT NUMBER

TR 10,689

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Science Foundation
1800 G St. NW
Washington, DC 20660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Ionosonde data were taken from two high-latitude Arctic radio links in order to investigate two-hop high-frequency (HF) polar propagation. A comparison of the two-hop data measurements with both IONCAP and ASAPS predictions has validated the accuracy of the model for complex skywave radio links.

14. SUBJECT TERMS

Antarctica, High-Frequency Communications, IONCAP, Ionosonde, Ionospheric Propagation, Radio Link, Skyway Link, Telecommunications, Transpolar Link, Two-Hop Link

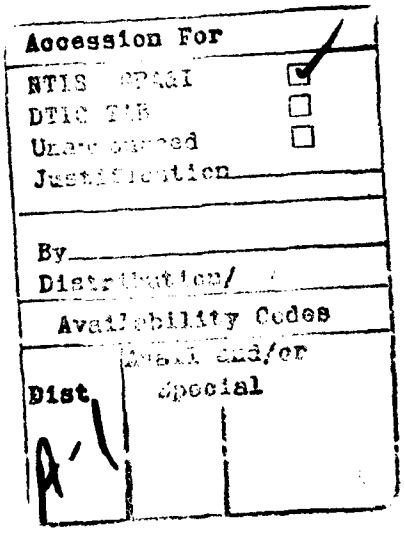
15. NUMBER OF PAGES

130

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

TABLE OF CONTENTS



LIST OF ILLUSTRATIONS

THE USE OF TWO-STATION IONOSONDE DATA FOR IONCAP/ASAPS MODEL VALIDATION ON TRANSPOLAR TWO-HOP, HIGH-FREQUENCY SKYWAVE RADIO LINKS

INTRODUCTION

BACKGROUND

An investigation* of one-hop, high-frequency (HF) trans-Arctic skywave propagation was conducted through a comparison of Scott Base (Antarctic) ionosonde measured data with a stand-alone prediction system (ASAPS) from the Australian Ionospheric Prediction Service and with IONCAP from the Institute for Telecommunication Services of the U.S. Department of Commerce. During this study, a FORTRAN program called IONOLINK (using the modern format for ionosonde data) was produced for the creation of oblique frequency availability.

The results of the investigation showed that both the IONCAP and ASAPS models predicted the single-hop propagation-dependent frequency of optimum transmission (FOT), maximum usable frequency (MUF), and highest propagation frequency (HPF)—frequencies not exceeded 10, 50, and 90 percent of the time, respectively—within 2 MHz for a wide range of sunspot numbers and solar zenith angles. (No attempt was made to determine the lowest usable frequency (LUF) values because they are not only dependent on propagation but also on the radio-link power budget.) The close agreement of model predictions with the measurements can reasonably be explained by the probable contributions of the Scott Base ionosonde data to the development of ionospheric electron density profiles used in both the ASAPS and IONCAP programs.

SCOPE

In this report, the one-hop results from the earlier study have been extended to the two-hop case using ionosonde data simultaneously acquired from two high-latitude (Arctic) stations. Although ionosonde data from a single station permits usable oblique frequency extrapolation from the vertical incidence measurements to any single-hop range, the use of two-station ionosonde data restricts the results to a single-hop range for each pair of ionosondes used. In addition, the ionosonde locations must be chosen so that their separation distance excludes one-hop

* Submarine Electromagnetic Systems Department, *High Frequency (HF) and Meteor Burst Communications in a Polar Environment*, NUWC-NPT Technical Document 10,375, Naval Undersea Warfare Center Detachment, New London, CT, 30 September 1993.

propagation, while the dominant mode is limited to two-hop propagation (i.e., the range should exceed about 4000 km but not be much greater than 8000 km). The comparison of IONCAP and ASAPS predictions with these two-hop frequency-availability values provides a more accurate validation of the model for complex links than could the one-hop cases of the previous study.

APPROACH

IONOSONDE DATA ANALYSIS

Northern Latitude Ionosonde Data

Ionosonde data measured at several Russian and Canadian ionosonde stations from 1991 were available for the characterization of local ionospheric skywave propagation conditions. The data consisted of monthly tables of hourly values for several scaled parameters measured from ionosonde output traces. One data file was provided for each scaled parameter for one or more months during 1991. These parameter files included the critical, or maximum, frequency f_v (MUF) for which the ionosphere would reflect a vertically launched radiowave from one or more ionospheric layers. Ideally, the virtual reflection height h' would also be provided or, at a minimum, the scale factor for determining the basic MUF value for single-hop propagation to a fixed distance. The F2, F1, and E layers define the most important ionospheric altitude bands used for HF skywave radio and therefore are of significant importance in the analysis of ionosonde parameters as depicted in figures 1 and 2. For the F2 layer, a MUF factor, M3000F2 (M_{F2}), is provided to convert the F2-layer critical frequency, foF2 (where o defines the ordinary wave), into the corresponding MUF values for a single-hop range of 3000 km. Note that MUF refers to the maximum usable oblique frequency at a given time, not the 50-percent value from the distribution. At this point, we will refer to this "basic MUF" value as the maximum propagating frequency extrapolated to a specified range from the midpath ionosonde data and let the "unmodified MUF" refer to the 50-percentile value.

The ionosonde data for each month and scaled parameter were presented in a single block and stored as a separate file. These files had to be merged to create a single file containing all the parameters scaled for each of the available months in order to produce the necessary input file. Selected files were merged to create input files for the IONOLNK2 computer program, which is a FORTRAN software analysis package developed to determine two-hop frequency availability using scaled ionosonde data from two stations. Ionosonde stations were selected to provide a total two-hop path length between 4000 and 8000 km and a second path length above 8000 km (the nominal upper bound on two-hop path length). In each file, a block header specified the ionosonde site, the date, and the identity of the scaled parameter whose values immediately followed the header. Each scaled parameter in a block consisted of five character positions—three positions carried an integer number and the other two, a descriptive and a qualifying letter. The scaled frequency values were presented in the data file in units of $10f_v$ MHz, with the heights given directly in kilometers and the basic MUF factors presented as $100M_f$. Many hours of various

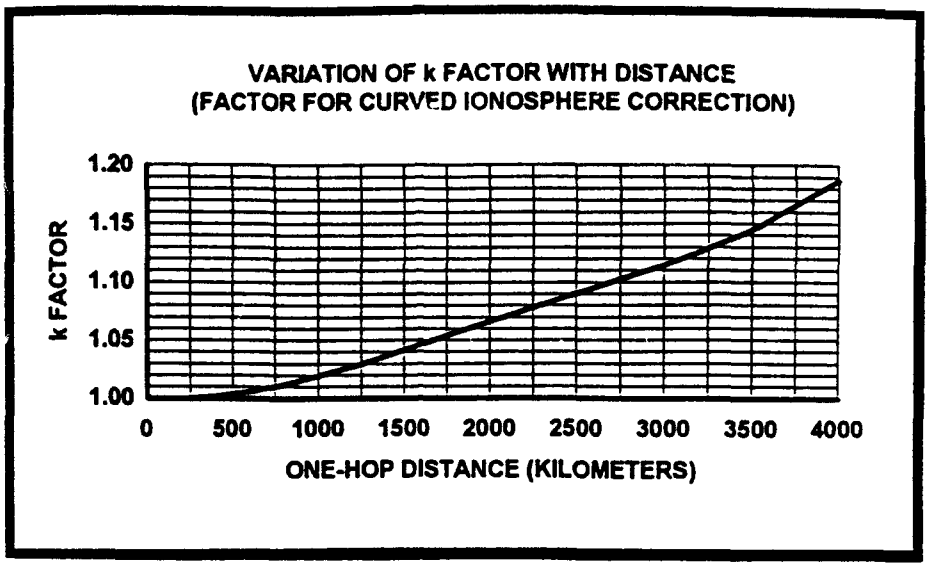


Figure 1. The *k*-Value Versus Range *D*

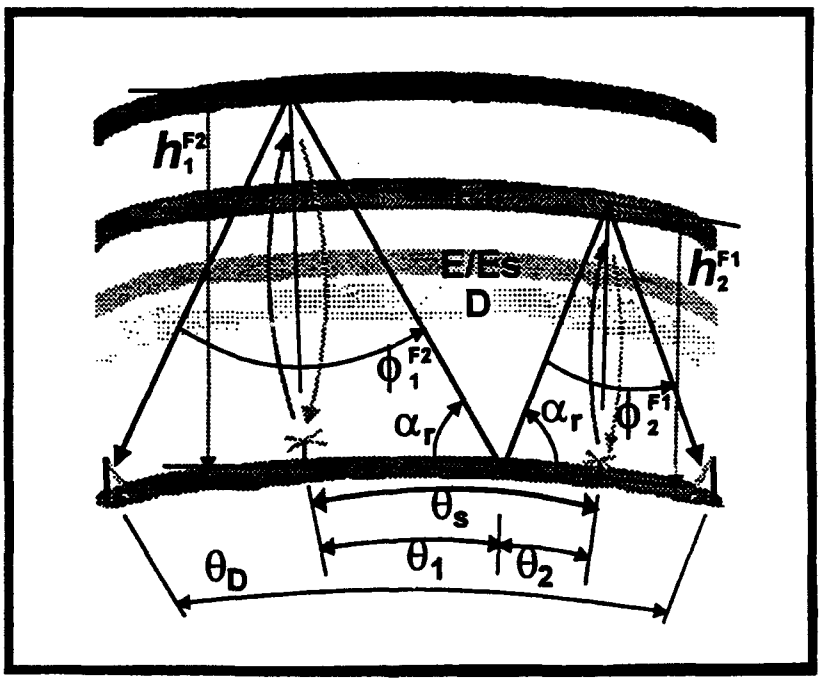


Figure 2. Two-Hop Link Geometry

months and years showed no parameter values or explanatory letter. A complete description of the data format is provided in appendix A.

OBLIQUE INCIDENCE

One-Hop Extrapolation

Let f_v be the ionosonde-reported critical frequency for a vertically incident signal from some ionospheric layer at a virtual height h'. The corresponding basic MUF value, f_{ob}, for single-hop oblique propagation over a link with great circle path (GCP) length D and the path midpoint located above the ionosonde may be given by

$$f_{ob} = k(D) f_v \sec(\phi_o) , \tag{1}$$

where the angle of incidence of the signal incident on a curved ionospheric layer is

$$\phi_o = \tan^{-1}\left[\frac{\sin(\theta_D/2)}{1+(h'/R_E)-\cos(\theta_D/2)}\right] \tag{2}$$

and

$$\theta_D = \frac{D}{R_E} . \tag{3}$$

In this equation, R_E is the earth's radius and $k(D)$ is the secant correction factor that approximates the effects of ionospheric curvature. The geometry represented in these expressions is depicted in figure 3. The function $k(D)$ is plotted in figure 1.

The elevation angle θ_{el} is measured from the horizon to the virtual reflection point and provides a means of estimating the usable portion of an antenna pattern during link design studies. From figure 3, this angle can be determined as

$$\theta_{el} = 180 - \alpha - \theta_D/2 , \tag{4}$$

where

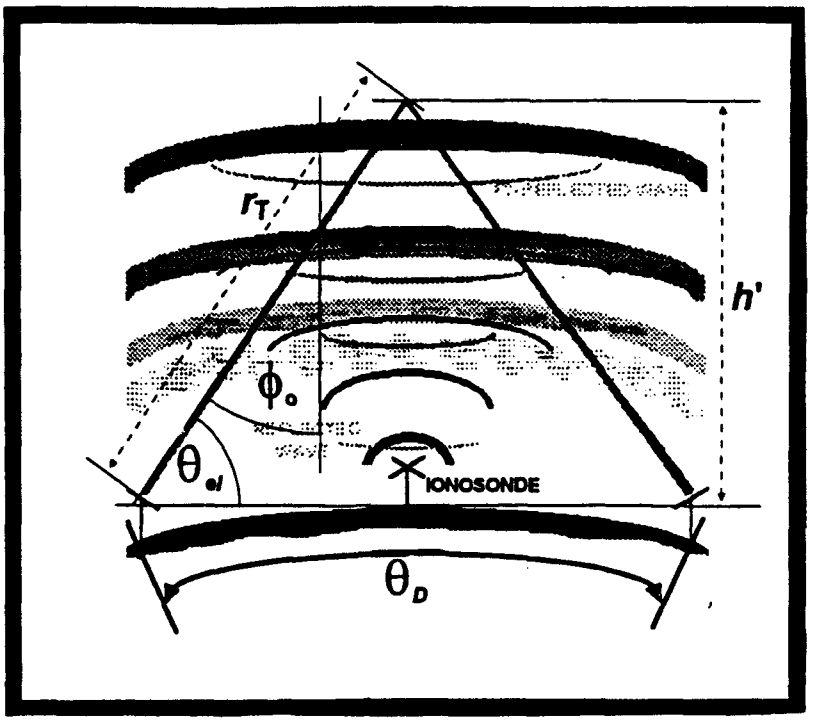


Figure 3. The HF Link Geometry

$$\alpha = \cos^{-1}\left[\frac{r_T^2 + (R_E + h')^2 - R_E^2}{2r_T(R_E + h')}\right] \tag{5}$$

and

$$r_T = \sqrt{h'^2\, 2R_E[1-\cos(\theta_D/2)](R_E + h')}\ . \tag{6}$$

When the virtual height is not available in the ionosonde data base, then it may either be computed from the corresponding basic MUF factor, M_f, or the average height of the layer (e.g., 325 km for the F2 layer). If the basic MUF factor is available, then the virtual height may be approximated by

$$h' = R_E\left[\frac{\sin(\theta_D/2)}{\tan(\phi_o)} + \cos(\theta_D/2) - 1\right], \tag{7}$$

where it is necessary to compute $\tan(\phi_o)$ from

$$\tan(\phi_o) = \frac{M_f \sqrt{1 - [k(D)/M_f]^2}}{k(D)} \quad . \tag{8}$$

These equations are used by the IONOLNK2 computer program (see appendix B) to determine the critical frequencies and elevation angles for one-hop oblique propagation with a path midpoint (reflection point) above each ionosonde station along the two-hop path.

For each hour for which the ionosonde data include a legitimate scaled value (i.e., for which a value is provided), the IONOLNK2 program first computes the F2-layer basic MUF value for each range D that is input by the user (i.e., $f_{F2}(D)$). Next, it computes $f_{F1}(D)$ and $f_E(D)$. Since near-vertical incidence skywave (NVIS) propagation may include two-hop propagation from the E layer, IONOLNK2 also computes the two-hop value $f_{2E}(D/2)$. Finally, the sporadic E layer (E_s) is considered for the computation of the one-hop value $f_{Es}(D)$ and the two-hop value $f_{2Es}(D/2)$. The basic MUF value, or link frequency f_L, for the given hour is then determined from

$$f_L = \text{MAX}\{f_{F2}, f_{F1}, f_E, f_{2E}, f_{Es}, f_{2Es}\} \quad . \tag{9}$$

The f_L values are used to increment cumulative distribution bins, which are normalized by the total number of usable ionosonde data hours in the chosen sample set.

Two-Hop Extrapolation

Consider the two-hop HF propagation geometry shown in figure 2. Two ionosonde stations are shown spaced by the earth-centered GCP angle θ_s. The two ionospheric reflections above these stations occur at virtual reflection heights h_1 and h_2, as shown in the figure. The corresponding "included" angles ϕ_1 and ϕ_2, respectively, must be determined to compute the desired MUF factors (see equation (1)). To find these angles, the location of the ground reflection point between the two ionosonde stations must first be found. This reflection point cannot be determined analytically, but may be solved for numerically by using a root-finding algorithm. Root-finding algorithms begin with an estimate of the desired root (reflection point) and then proceed to minimize some error function determined from the subsequent root estimates. In this case, it is known that the angle of incidence (α_r) equals the angle of reflection. Given θ_1 and $\theta_2 = \theta_s - \theta_1$, two estimates of the angle of ground reflection ($\hat{\alpha}_{r1}$ and $\hat{\alpha}_{r2}$) can be computed and used in the error function as follows:

$$\varepsilon = \hat{\alpha}_{r1} - \hat{\alpha}_{r2} \quad , \tag{10a}$$

where

$$\hat{\alpha}_{ri} = \sin^{-1}\left(\frac{r_i^2 - a_i^2 - R_e^2}{2R_e a_i}\right), \tag{10b}$$

with

$$a_i = \sqrt{r_i^2 + R_e^2 - 2r_i R_e \cos(\theta_i)} \tag{10c}$$

and

$$r_i = h_i + R_e, \quad \text{for } i = 1,2. \tag{10d}$$

The root-finding algorithm attempts to select θ_1 such that $|\epsilon|$ is less than some small positive number. Given an estimate θ_1, the ground distance covered by the total two-hop path is then given by

$$D = 2(\theta_1 + \theta_2) * R_E . \tag{11}$$

Thus, the use of ionosonde data from two stations to characterize the overhead conditions for ionospheric reflection restricts the extrapolation to a single two-hop length. However, the actual endpoints of the link will change as the reflection heights change. If the ionospheric conditions in the vicinity of an ionosonde are considered identical over the sky volume in which the corresponding ionospheric reflections occur, then the relative frequency availability distribution for a fixed-length link should be insensitive to exact geographic location. This assumption is necessary to justify the empirical calculations of two-hop frequency availability between fixed points using the IONCAP or ASAPS computer program.

There are 16 possible two-hop propagation modes considered in this analysis: F2-F2, F2-F1, F2-E, F2-Es, F1-F2, F1-F1, F1-E, F1-Es, E-F2, E-F1, E-E, E-Es, Es-F2, Es-F1, Es-E, and Es-Es, where the hyphen (-) signifies ground reflection. The IONOLNK2 program, mentioned earlier, was designed to determine the monthly two-hop FOT, MUF, and HPF values as well as frequency and mode density functions over the entire processed time interval. This program inputs ionosonde data from two stations, determines the resulting two-hop oblique propagation geometry, and then calculates the basic MUF value for each of the 16 possible modes. Of course, links

exceeding about 4000 km will not experience two-hop E-layer modes because of the lower reflection heights as compared with the F-layer modes. IONOLNK2 then determines the minimum of the resulting basic MUF values as the basic two-hop MUF value. This value is used to increment the appropriate probability bins. The IONOLNK2 computer program is described in appendix A.

Reverse Blank

ANALYSIS OF RESULTS

DIKSON (RUSSIA) AND RESOLUTE BAY (CANADA)

Scaled ionosonde parameter files from the Dikson (No. 373, 72.5° N and 80.4° E) and the Resolute Bay (No. 974, 74.7° N and 95.0° W) ionosonde stations were used to produce a 7100-km two-hop HF path. Although 12 months of scaled parameter data were taken, only 6 months of Resolute Bay data were provided for this study. By merging the appropriate data files (see appendix A), a single ionosonde file was created for each station for January through June 1991. Figure 4 is a plot of the frequency density function covering the entire 6-month period for which data from both stations were available. The values plotted in the figure represent the percentage of days during the 6 months when simultaneous dual-station ionosonde data from Dikson and Resolute Bay could be used to extrapolate to two-hop oblique propagation. The small number of usable hours is due to lack of parameter scaling at one or both sites and to the dominant E-layer reflections that prevent adequate two-hop range. The month-by-month comparison of ionosonde-derived frequency-availability values in the form of FOT, MUF, and HPF versus time-of-day are presented in appendix B (pages B-3 to B-20). Both IONCAP and ASAPS computer model predictions are plotted in these figures along with the ionosonde-extrapolated results. Overall, the predicted values show less diurnal variation than the corresponding measurement-derived values, as shown in the MUF comparison for February on page B-7.

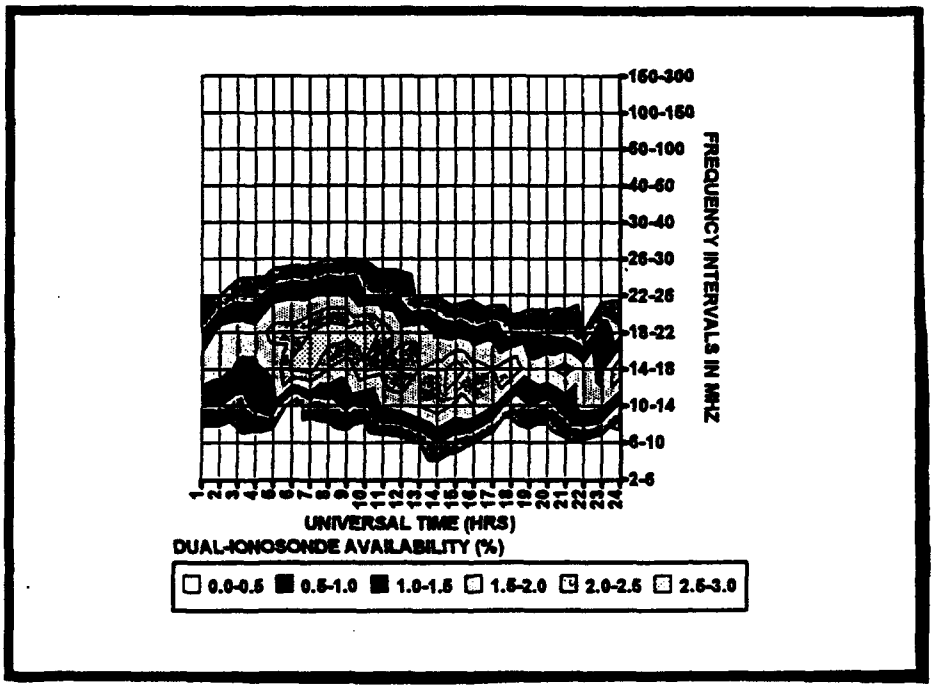


Figure 4. Six-Month Frequency Availability for Dikson and Resolute Bay

CAPE SCHMIDT (RUSSIA) AND LOPARSKAYA (RUSSIA)

A second set of ionosonde stations was selected to form a 9100-km two-hop trans-Arctic skywave path from Cape Schmidt (No. 669, 68.8° N and 179.5° E) to Loparskaya (No. 168, 68.0° N and 33.0° E) in Russia. A full 12 months of ionosonde data from both stations were available for empirical extrapolation, although only March, July, September, and December were chosen for the month-by-month comparison with the prediction. The longer link range significantly reduced the number of hours in which the extrapolation yielded two-hop propagation, as shown in figure 5. The corresponding measurement-prediction comparisons are presented in appendix B (pages B-22 to B-33). Very few of the IONCAP and none of the ASAPS predictions demonstrated two-hop propagation at this range.

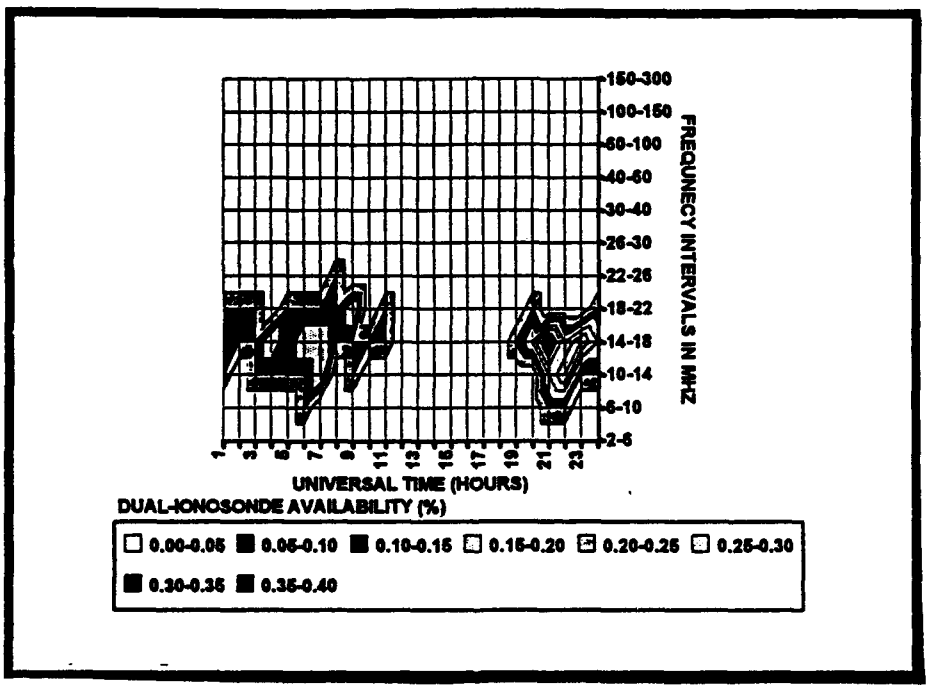


Figure 5. Twelve-Month Frequency Availability for Cape Schmidt and Loparskaya

CONCLUSIONS AND RECOMMENDATIONS

This report has summarized work performed to extend the extrapolation of ionosonde measurements from the single-hop oblique geometry to the two-hop case. A computer program called IONOLNK2 was developed from the one-hop program IONOLINK to permit FOT, MUF, and HPF comparisons with IONCAP and ASAPS predictions.

The IONOLNK2 program should be used for mid-latitude two-hop paths in which more simultaneous data are available and fewer E-layer reflections are observed. This effort should be performed with both the IONOLINK and IONOLNK2 programs to provide a complete characterization of frequency availability for the one- and two-hop propagation paths, respectively. Once the investigation is completed, and the relationship between empirical extrapolation and computer prediction is firmly established for mid-latitude paths, an extensive high-latitude study encompassing more years of ionosonde data should be performed. This high-latitude effort should include use of the ICECAP computer model, so that a comparison of its predictions with one- and two-hop extrapolation as well as with IONCAP/ASAPS predictions may be made.

APPENDIX A
IONOLNK2 COMPUTER PROGRAM

A.1 DESCRIPTION AND INSTALLATION

IONOLNK2 is a Microsoft FORTRAN 5.0 computer program developed and executed on a Gateway 2000 486/33-MHz PC. The program was written and debugged using the Power Workbench (PWB) development tool from Microsoft (MS) which is available with its FORTRAN product. MS-FORTRAN 5.0 employs ANSI FORTRAN 77 as well as several additional features for compatibility with VAX FORTRAN available from the Digital Equipment Corporation. The purpose of the IONOLNK2 program is to accumulate numerical densities for two-hop oblique standard maximum usable frequency (MUF) values using input files in the "NEW URSI" data format as published in International Council of Scientific Unions Panel on World Data Centers "GUIDE to the WORLD DATA CENTER SYSTEM", Part 2. These parameters include hourly junction frequencies (JFs), "standard" maximum usable frequency (MUF) multipliers, and ionospheric reflection heights (virtual heights), for the each of the observed ionospheric layers.

IONOLNK2 consists of a two source code modules, IONOLNK2.FOR and HFLINK2.FOR which are compiled and linked by the PWB tool using the "make" file called IONOLNK2.MAK into IONOLNK2.EXE. Proper use of the MB-PWB tool is beyond the scope of this appendix. The executable image is invoked at the user prompt from drive C by the command:

C>**IONOLNK2**

where the highlighted and underlined text indicates user entry. The "C>" in this example is displayed by the PC. The program then asks the user for the name of the desired IONOLNK2 input file with the following prompt:

Type name of input file without extension ".INP"

The user then types the name of the input file *without* the trailing (.) or file extension. The IONOLNK2 program *assumes* that the input file extension is ".INP". For example, if the user's input file is called DKRS1991.INP and it resides in directory "INPUT" on drive "D", then the user would respond to the prompt with

D:\INPUT\DKRS1991

and the IONOLNK2 program would automatically read file DKRS1991.INP in the indicated directory.

A.2 INPUT FILE DESCRIPTION

A.2.1 IONOLNK2 Input File

The user-input file must be created using a standard ASCII editor. IONOLNK2 reads this input file in a format-free format, so exactly one comma (,) or one or more spaces may be used to delimit entries. For example, consider the following example input file:

```
'DIKSON', 'RESOLUTE BAY'                                                    1
1, 1, 1, 1                                                                  2
24, 31, 14, 2000                                                            3
T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T                           4
T,T,T,T,T,T,T,T,T,T,T,T,T                                                   5
F                                                                           6
T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T      7
T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T,T      8
10, 20.0, 40.0, 60.0, 80.0, 100.0, 200.0, 400.0, 800.0, 1000.0, 2000.0      9
2.0, 32.0, 60                                                               10
100.0, 4000.0, 60                                                           11
0.0, 90.0, 60                                                               12
1, T, 0.0, 12.0, 60                                                         13
2, F, 0.0, 30.0, 60                                                         14
3, F, 0.0, 30.0, 60                                                         15
4, T, 2.0, 4.0, 50                                                          16
5, F, 0.0, 600.0, 60                                                        17
6, F, 0.0, 600.0, 60                                                        18
7, T, 0.0, 60.0, 60                                                         19
8, F, 0.0, 30.0, 60                                                         20
9, F, 0.0, 10.0, 50                                                         21
10, F, 0.0, 600.0, 60                                                       22
11, T, 150.0, 600.0, 45                                                     23
12, T, 0.0, 5.0, 25                                                         24
13, F, 1.0, 30.0, 60                                                        25
14, T, 90.0, 150.0, 60                                                      26
15, F, 0.0, 600.0, 60                                                       27
16, T, 0.0, 10.0, 50                                                        28
17, F, 1.0, 30.0, 60                                                        29
18, T, 0.0, 5.0, 25                                                         30
19, F, 1.0, 30.0, 60                                                        31
20, T, 90.0, 190.0, 20                                                      32
21, F, 1.0, 30.0, 60                                                        33
22, T, 0.0, 5.0, 25                                                         34
23, F, 0.0, 10.0, 50                                                        35
24, F, 0.0, 600.0, 60                                                       36
25, F, 1.0, 30.0, 60                                                        37
26, T, 0.0, 16.0, 40                                                        38
27, F, 1.0, 30.0, 60                                                        39
28, F, 1.0, 30.0, 60                                                        40
29, F, 1.0, 30.0, 60                                                        41
30, F, 1.0, 30.0, 60                                                        42
F                                                                           43
'D:\NUWCION\DK37391.NEW', 'D:\NUWCION\RS97491.NEW'                          44
```

Each record (line) is shown numbered to the right of the input values to provide an index for reference to each item in this description. These numbers must *not* appear in the user-input file or an input error will occur. A description of each input value referenced by these numbers is provided in Table A-1.

Table A.1 Input Variable Descriptions for IONOLNK2 Program

INPUT FILE ROW NUMBER	ITEM NUMBER IN ROW	FORTRAN VARIABLE TYPE	DESCRIPTION	TYPICAL VALUES
1	1, 2	CHARACTER*20	Names of two ionosonde stations used to find the start of each month's data records	see example file
2	1	INTEGER*4	First hour in time interval for bin-accumulation	1
"	2	"	First day in time interval for bin accumulation	1
"	3	"	First month in time interval for bin accumulation	1
"	4	"	First year in time interval for bin-accumulation	1970
3	1	INTEGER*4	Last hour in time interval for bin-accumulation	24
"	2	"	Last day in time interval for bin accumulation	31
"	3	"	Last month in time interval for bin accumulation	12
"	4	"	Last year in time interval for bin-accumulation	1972
4	1-25	LOGICAL*1	Use/ignore MASK for up to 25 years of ionosonde data files, one file per year, e.g., 1970 - 1972	T or F for each entry
5	1-12	LOGICAL*1	Use/ignore MASK for up to 12 months of ionosonde data for each year (file), 1 through 12	T or F for each entry
6	1	LOGICAL*1	Logic switch for program IONOLNK2, not used in program IONOLNK2: T creates hourly FOT, MUF (standard), and HPF statistics for oblique HF links, F creates same density tabulations	T or F ignored
7	1-31	LOGICAL*1	Use/ignore MASK for up to 31 days of ionosonde data for each month (1 through 31)	T or F for each entry
8	1-24	LOGICAL*1	Use/ignore MASK for up to 24 hours of ionosonde data for each day (1 through 24)	T or F for each entry

Table A.1 Input Variable Descriptions for IONOLNK2 Program (Continued)

INPUT FILE ROW NUMBER	ITEM NUMBER IN ROW	FORTRAN VARIABLE TYPE	DESCRIPTION	VALUES
9	1	INTEGER*4	Input to track IONOLINK [] input files but *not* used in IONOLNK2 (10 MAX)	10
9	2-11	REAL*4	Oblique path lengths (km) for IONOLINK [], *not* used in program IONOLNK2 (10 values MAX)	0.0 to 4000.0
10	1	REAL*4	Lower frequency limit (MHz) for IONOLNK2-computed FOT, MUF, & HPF values	2.0
"	2	REAL*4	Upper frequency limit (MHz) for IONOLNK2-computed FOT, MUF, & HPF values	60.0
"	3	INTEGER*4	Number of frequency bins for IONOLNK2-computed FOT, MUF, & HPF values (60 MAX)	56
"	4	LOGICAL*1	Logical switch to turn off sporadic E layer in IONOLINK [], *not* used in IONOLNK2	T or F
11	1	REAL*4	Lower height limit (km) for IONOLNK2-computed FOT, MUF, & HPF values in IONOLINK [], *not* used in IONOLNK2	0.0
"	2	REAL*4	Upper height limit (MHz) for IONOLNK2-computed FOT, MUF, & HPF values in IONOLINK [], *not* used in IONOLNK2	600.0
"	3	INTEGER*4	Number of height bins for IONOLNK2-computed FOT, MUF, & HPF values (60 MAX) in IONOLINK [], *not* used in IONOLNK2	60

Table A.1 Input Variable Descriptions for IONOLNK2 Program (Continued)

INPUT FILE ROW NUMBER	ITEM NUMBER IN ROW	FORTRAN VARIABLE TYPE	DESCRIPTION	VALUES
12	1	REAL*4	Lower antenna elevation angle limit (deg) for IONOLNK2-computed FOT, MUF, & HPF values	0.0
"	2	REAL*4	Upper antenna elevation angle limit (deg) for IONOLNK2-computed FOT, MUF, & HPF values	90.0
"	3	INTEGER*4	Number of angle bins for IONOLNK2-computed FOT, MUF, & HPF values (60 MAX)	60
13, 14, 15, 19, 20, 24, 25, 28, 29, 30, 31, 33, 34, 37, 38, 39	1	LOGICAL*1	Logical switch to turn on analysis for the corresponding ionosonde-scaled-frequency parameter codes described in Section A.2.2. A "T" tells the IONOLNK2 program to process the parameter, not processed if "F".	T or F
"	2	REAL*4	Minimum value of frequency to be used in density accumulation bins	2.0
"	3	REAL*4	Maximum value of frequency to be used in density accumulation bins	30.0
"	4	INTEGER*4	Number of frequency bins to be used in density accumulation bins	28
16, 21, 35, 40	1	LOGICAL*1	Logical switch to turn on analysis for the corresponding ionosonde-scaled-factor parameter codes described in Section A.2.2. A "T" tells the IONOLNK2 program to process the parameter, not processed if "F".	1 T or F
"	2	REAL*4	Minimum value of factor to be used in density accumulation bins	2.0
"	3	REAL*4	Maximum value of factor to be used in density accumulation bins	4.0
"	4	INTEGER*4	Number of factor bins to be used in density accumulation bins	20

Table A.1 Input Variable Descriptions for IONOLNK2 Program (Continued)

INPUT FILE ROW NUMBER	ITEM NUMBER IN ROW	FORTRAN VARIABLE TYPE	DESCRIPTION	VALUES
17, 18, 22, 23, 26, 27, 32, 36	1	LOGICAL*1	Logical switch to turn on analysis for the corresponding ionosonde-scaled-height parameter codes described in Section A.2.2. A "T" tells the IONOLNK2 program to process the parameter, not processed if "F".	T or F
"	2	REAL*4	Minimum value of height to be used in density accumulation bins	0.0
"	3	REAL*4	Maximum value of height to be used in density accumulation bins	600.0
"	4	INTEGER*4	Number of height bins to be used in density accumulation bins	60
41, 42	1	LOGICAL*1	Logical switch to turn on analysis for the corresponding ionosonde-scaled parameter code as described in Section A.2.2. A "T" tells the IONOLNK2 program to process the parameter, not processed if "F".	T or F
"	2	REAL*4	Minimum value of parameter to be used in density accumulation bins	0.0 (?)
"	3	REAL*4	Maximum value of parameter to be used in density accumulation bins	600.0 (?)
"	4	INTEGER*4	Number of parameter bins to be used in density accumulation bins	60 (?)
43	1	LOGICAL*1	Logical switch to have densities on total elapsed time or elapsed time for which ionosonde data was possible, i.e., no equipment failure (qualifying letter "C")	T or F
44-68	1	CHARACTER*20	Ionosonde data filename pairs, including drive and subdirectory, with up to 25 filename pairs allowed	See Section A.2.1 for examples

Table A.2 Ionosonde (New URSI) Data Format

Record	Columns	Description
1	1-20	Station name
1	21-25	Station code
1	26-29	Standard time meridian of the station (e.g. 150W, 90E, etc., with 000W or 000E = UT)
1	30- 33	Geographic co-latitude in tenths of a degree
1	34- 37	Geographic East longitude in tenths of a degree
1	38- 41	Year
1	42- 43	Month
1	44- 45	Parameter code (See UAG-23 characteristic code)
1	46-120	Spare
2	24 X 5-character code	Hourly data for the first day of the month (i.e. foF2 values - 078 R, 080 , etc.)
3	24 X 5-character code	Hourly data for the second day of the month (i.e. foF2 values - 078 R, 080 , etc.)
.	24 X 5-character code	.
.	24 X 5-character code	.
.	24 X 5-character code	.
32	24 X 5-character code (if available)	Hourly data for the thirty-first day of the month (If less than 31 days, blank fill.)
33		Medians
34		Median Count
35		Upper Quartile
36		Lower Quartile
37		Upper Decile
38		Range
39		Lower Decile
40		Spare

Not all of the five-character scaled-parameter values were reported for each hour of the ionosonde data provided for the study. In this case, blank fill was used as a place holder. Note that the files used in this study were merged from separate parameter-month files to form complete parameter files for all available months. The completed ionosonde data files were named CS66991.NEW, DK37391.NEW, LY16891.NEW, and RS97491.NEW for Cape Schmidt, Dikson, and Loparskaya in Russia and Resolute Bay in Canada, respectively. None of these files included rows 33-40 in the monthly data blocks for each parameter. If N parameters have been scaled, then there will be N blocks for each of the months contained within a single file. Also, each pair of files used in the two-hop IONOLNK2 analysis must contain exactly the same parameter-month blocks or program execution will terminate. Thus, "null" blocks may have to be inserted into files that include the first record (see Table A.2) followed by 31 rows of 24 "equipment fault" entries given by " C ". For the standard N = 14 parameters, the ionosonde data file should be organized as shown in Table A.3. A complete description of the parameter codes, descriptive, and qualification letters summarized in Table A.4a, b, and c, respectively, can be found in UAG-23.

Table A.3 Ionosonde Data File Organization

Block	Month	Parameter (Characteristic)	Numeric code
1	January	foF2	00
2	January	M(3000)F2	03
.	.	.	.
.	.	.	.
.	.	.	.
13	January	fxI	51
14	January	fmI	52
15	February	foF2	00
16	February	M(3000)F2	03
.	.	.	.
.	.	.	.
.	.	.	.
167	December	fxI	51
168	December	fmI	52

Table A.4a Parameter Codes

Description	Parameter Codes and Meaning					
F2 layer	00-foF2	01-fxF2	02-fzF2	03-M(3000)F2	04-h'F2	05-hpF2
F1 layer	10-foF1	11-fxF1	13-M(3000)F1	14-h'F1	16-h'F	
E layer	20-foE	22-foE2	24-h'E	26-h'E2		
Es layer	30-foEs	31-fxEs	32-fbEs	33-fEs	34-h'Es	
Other	40-foF1.5	42-fmin	43-m(3000)F1.5	44-h'F1.5		
Spread	50-foI	51-fxI	52-fmI			
TEC	70-I(2000)	71-I	72-I(xxx)			

Table A.4b Qualifying Letters

Letter	Meaning
A	Less than (used only in case of total blanketing)
D	Greater than
E	Less than
I	Interpolated
J	Deduced from x component
M	Mode uncertain
O	Deduced from o component
T	Smoothed from sequence
U	Uncertain
Z	Deduced from z component

Table A.4c Descriptive Letters

Letter	Meaning
A	Blanketing
B	Absorption
C	Non-ionospheric (equipment)
D	Above upper freq. range
E	Below lower freq. range
F	Spread echoes
G	Ionization density too small
H	Stratification
K	Night E layer present
L	Insufficiently defined cusp
M	Mode uncertain
N	Superimposed layers
O	Measurement refers to o component
Q	Range spread
R	Attenuation near critical freq.
S	Interference
T	Interpolated
V	Forked trace
W	Above height range
X	Measurement refers to x component
Y	Lacuna (tilt)
Z	Measurement refers to z component

A.3 OUTPUT FILE DESCRIPTION

The IONOLNK2 computer program can generate two different output file formats depending on the value of input file record number six (6). In either case, the output file begins with a description of the two-hop link geometry, including the geographic coordinates of the two ionosonde stations. If record six is "false" ("F"), the IONOLNK2 output filename has the same filename as the input file but with the extension ".OUT" instead of ".INP". It contains a tabular normalized number density (probability density function) for each possible propagation mode (currently one of the 16 possible two-hop modes F2-F2, F2-F1, F2-E, F2-Es, F1-F2, F1-F1, F1-E, F1-Es, E-F2, E-F1, E-E, E-Es, Es-F2, Es-F1, Es-E, or Es-Es), the average standard MUF value, and the standard deviation of the standard MUF value for each hour of the day. These statistics are created over the entire time interval requested by the input file. The format for these density tables is depicted in Tables A.5a and A.5b. A sample output file called DKRS1991.OUT (Dikson-Resolute Bay) is provided with the software package.

The second output file format was developed to provide a direct output for comparison with IONCAP, ASAPS, or other HF prediction code output. This output file provides the optimum working frequency (OWF or FOT, 10% not exceeded), the median standard MUF value (50% not exceeded), and the lower decile standard MUF value (90% not exceeded) versus hour of day. In addition, the corresponding average antenna take-off angle for each standard MUF-distribution value is also provided. This output file has the extension ".LNK" and cannot be created simultaneously with the

corresponding ".OUT" file. Parameter six (6) in the input file determines the form of IONOLNK2 output. The format of this file is outlined in Table 6 and a sample "*.LNK" file is provided with the software.

Table A.5a Propagation Mode Density Table

	Hour Intervals (UT)							
PROPAGATION MODE: ...	0-1	1-2	2-3	.	.	.	23-24	NET
100 Pr	*nnn*[1]	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*
AVGFRQ	*fff*[2]	*fff*	*fff*	*fff*	*fff*	*fff*	*fff*	*fff*
SDVFRQ	*sss*[3]	*sss*	*sss*	*sss*	*sss*	*sss*	*sss*	*sss*

[1]*nnn* is a probability expressed as a percentage ($0 \le nnn \le 100$)
[2]*fff* is a frequency (MHz) [3]*sss* is frequency standard deviation (MHz)

Table A.5b Standard-MUF Density Table

	Hour Intervals (UT)							
Frequency (MHz)	0-1	1-2	2-3	.	.	.	23-24	NET
f_1-f_2*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*
f_2-f_3	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*
f_3-f_4	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*
.	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*
.	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*
.	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*
f_{N-1}-f_N	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*	*nnn*

*f_i - f_{i+1} implies that the standard MUF value f is within the interval $f_{i-1} < f \le f_i$ with the indicated probability.

Table A.6 FOT, MUF, HPF Table in "*.LNK" Output File for Each Month

10% NOT EXCEEDED (FOT)			*year*	*month*				
(HOUR:)	0-1	1-2	2-3	.	.	.	23-24	NET
FOT:	*fff*[1]	*fff*	*fff*	*fff*	*fff*	*fff*	*fff*	*fff*
ANG:	*aaa*[2]	*aaa*	*aaa*	*aaa*	*aaa*	*aaa*	*aaa*	*aaa*
50% NOT EXCEEDED (MUF)			*year*	*month*				
(HOUR:)	0-1	1-2	2-3	.	.	.	23-24	NET
MUF:	*fff*	*fff*	*fff*	*fff*	*fff*	*fff*	*fff*	*fff*
ANG:	*aaa*	*aaa*	*aaa*	*aaa*	*aaa*	*aaa*	*aaa*	*aaa*
90% NOT EXCEEDED (HPF)			*year*	*month*				
(HOUR:)	0-1	1-2	2-3	.	.	.	23-24	NET
HPF:	*fff*	*fff*	*fff*	*fff*	*fff*	*fff*	*fff*	*fff*
ANG:	*aaa*	*aaa*	*aaa*	*aaa*	*aaa*	*aaa*	*aaa*	*aaa*

[1]*fff* - frequency in MHz [2]*aaa* - angle in degrees

A.3 IONOLNK2 PROGRAM LISTING

A.3.1 IONOLNK2.FOR Source Program

```
$FREEFORM
$LARGE

    INTERFACE TO SUBROUTINE HFLINK2( i1, i2, l1, l2 )
    INTEGER*4 i1, i2
    LOGICAL*1 l1(4,4), l2
    END

    INTERFACE TO SUBROUTINE PARMBN2( i1, i2, i3, i4, r1, c1, c2 )
    REAL*4 r1
    INTEGER*4 i1, i2, i3, i4
    CHARACTER*1 c1, c2
    END



"                   Program HFSOND2
"
" Purpose: Process an IONOSONDE data file following the conventions
"        established in UAG-23 and UAG-23A and determine the optimum
"        frequencies for an HF communications link with its midpath
"        above the ionosonde.
"
" Assumptions: It is assumed that both ionosonde data files have the
"           same number of parameter data blocks per month. If not,
"           then these input files must be modified to include a
"           "filler" block containing "  C " entries for the
"           31 rows of 24 scaled-parameter values.

    PROGRAM HFSOND2

" Declare variable types and dimension arrays

" Define a general input record with 120 columns and a holder for the
" previous record

    CHARACTER*120 RECORD_I(2), RECORD(2)

" Define parameters to hold values describing the ionosonde station

    CHARACTER*20 STATION_name(2), STATION_check(2)
    CHARACTER*5 STATION_code(2)
    CHARACTER*4 TIME_meridian(2), STATION_COLAT_ddg(2)
```

```
      CHARACTER*4 STATION_ELNG_ddg(2), YEAR(2)
      CHARACTER*2 MONTH(2), OLD_MONTH
      CHARACTER*6 f_RANGE(12)
      INTEGER*4 P_code(2), N_file, M_1, M_2, Y_1, Y_2, Y_3, Y_4
      INTEGER*4 YEAR_i, MONTH_i, DAY_i, HOUR_i
      INTEGER*4 YEAR_f, MONTH_f, DAY_f, HOUR_f

*     Define 5-character code derived from UAG-23 for hourly interval

      CHARACTER*1 P_char(3), Q_char, D_char

*     Define bins for yearly distribution of hourly values

      REAL*4 d_P, P_1, P_2, P_3, P_value, MAX_hours(1:30,1:24)
      REAL*4 P_low(1:30), P_hgh(1:30), P_scale(1:3)

      INTEGER*4 i_FILE, j_CODE(2), N_CODE, i_CODE, P_index(1:73), N_DISP
      INTEGER*4 N_Pbins(1:30), i_MONTH, PARM_code(2), P_type(1:30)
      INTEGER*4 ERR_LINE(2)

*     Declare arrays for HF link calculations

      REAL*4 fMHz_low, fMHz_hgh, dkm_low, dkm_hgh, Re_km, Rearth_km
      REAL*4 fMHz_hr_disp, fMHz_disp
      REAL*4 adg_low, adg_hgh, d_fMHz, d_dkm
      REAL*4 a_limit, adg_bin, adg_hr_bin, adg_mode_bin
      REAL*4 adg_mode_hr_bin, AVG_adg_mode_hr, AVG_adg_mode
      REAL*4 d_limit, dkm_bin, dkm_hr_bin, dkm_mode_bin, D_km(10)
      REAL*4 dkm_mode_hr_bin, AVG_dkm_mode_hr, AVG_dkm_mode
      REAL*4 f_limit, fMHz_bin, fMHz_hr_bin, fMHz_mode_bin
      REAL*4 fMHz_mode_hr_bin, AVG_fMHz_mode_hr, AVG_fMHz_mode
      REAL*4 SDV_fMHz_mode_hr, SDV_fMHz_mode, N_noLOS
      REAL*4 N_dy, N_hr_dy, N_mode_dy, N_mode_hr_dy, N_dy_hr, N_hr, N_GOOD
      REAL*4 COLAT_dg(2), LNG_dg(2), Ds_km, As_dg, azimuth_dg(2)

      INTEGER*4 i_DAY, i_HOUR, i_DST, i_FRQ, i_MODE, i_SITE
      INTEGER*4 N_FRQ, N_ANG, N_DST, N_Dkm
      INTEGER*4 N_bins, month_OLD

      CHARACTER*5 L_heading(3), MODE_name(1:4,1:4)
      CHARACTER*25 DIST_heading

*     Define storage for each parameter code, descriptive, and qualifying
*     letter.

      REAL*4 foF2, fxF2, fzF2, M3000F2, hF2, hpf2, foF1, fxF1
      REAL*4 M3000F1, hF1, hF, foE, foE2, hE, hE2, foEs
```

```
      REAL*4 fxEs, fbEs, fEs, hEs, foF1d5, fmin, M3000F1d5, hf1d5
      REAL*4 foI, fxI, fmI, I2000, II, Ixxx

*  Define unused logical parameter employed in the IONSTATS program

      LOGICAL*1 DIV_flag

*  Define a character string of BLANK_line for null string searches

      CHARACTER*21 BLANK_line

*  Define list of input files for analysis

      CHARACTER*25 S1_files(1:25), S2_files(1:25), INP_file
      CHARACTER*25 LNK_file, FILE_NAME, OUT_file, file_INP

*  Define heading variables for output

      CHARACTER*10 P_names(1:30)
      CHARACTER*5 QUAL_heading, DESC_heading
      CHARACTER*5 P_heading(1:3), DIM_heading(1:4)
      CHARACTER*25 HOUR_heading
      CHARACTER*1 QUAL_value(1:10), DESC_value(1:23)

      LOGICAL*1 YEAR_mask(1:25), MONTH_mask(1:12), DAY_mask(1:31)
      LOGICAL*1 HOUR_mask(1:24), CODE_mask(1:30), CODE_flag
      LOGICAL*1 P_flag, LINK_flag(4,4), L_flag
      LOGICAL*1 P10_FLAG, P50_FLAG, P90_FLAG
      LOGICAL*1 READ_flag, RUN_flag, Rept_flag, DAY_flag, HOUR_flag
      LOGICAL*1 MONTHLY_flag, FIRST_month, GEOG_flag

*  ********************************************************************
*  Define COMMON blocks

      COMMON/IONSNP/ foF2(2,31,24), M3000F2(2,31,24), hF2(2,31,24), -
                foF1(2,31,24), M3000F1(2,31,24), hF1(2,31,24), -
                hF(2,31,24), foE(2,31,24), -
                hE(2,31,24), foEs(2,31,24), fbEs(2,31,24), -
                    fEs(2,31,24), hEs(2,31,24)

*               fmin(2,31,24), fxI(2,31,24)

      COMMON /MISC/ Re_km, Rearth_km

      COMMON /LIMIT/ fMHz_low, fMHz_hgh, N_FRQ, dkm_low, dkm_hgh, N_DST, -
                adg_low, adg_hgh, N_ANG, LNK_file, d_fMHz, d_dkm, -
                d_adg, As_rd, Ds_km
```

```
*     COMMON /ANGLE/ adg_bin(0:61), -
*               adg_hr_bin(1:24,0:61), adg_mode_bin(1:4,1:4,0:61), -
*               adg_mode_hr_bin(1:4,1:4,1:24,0:61), -
*               AVG_adg_mode_hr(1:4,1:4,1:24), AVG_adg_mode(1:4,1:4), -
*               ANG_hr_frq(1:24,0:61,1:2)

      COMMON /ANGLE/ ANG_hr_frq(1:24,0:61,1:2)

*     COMMON /RANGE/ dkm_bin(0:61), -
*               dkm_hr_bin(1:24,0:61), dkm_mode_bin(1:4,1:4,0:61), -
*               dkm_mode_hr_bin(1:4,1:4,1:24,0:61), -
*               AVG_dkm_mode_hr(1:4,1:4,1:24), AVG_dkm_mode(1:4,1:4)

      COMMON /FRQBN/ f_limit(1:12,1:2), fMHz_bin(0:61), -
               fMHz_hr_bin(1:24,0:61), fMHz_mode_bin(1:4,1:4,0:61), -
               fMHz_mode_hr_bin(1:4,1:4,1:24,0:61), -
               AVG_fMHz_mode_hr(1:4,1:4,1:24), -
               SDV_fMHz_mode_hr(1:4,1:4,1:24), -
                    AVG_fMHz_mode(1:4,1:4), SDV_fMHz_mode(1:4,1:4), -
                    fMHz_hr_disp(1:24,1:12), fMHz_disp(1:12)

      COMMON /COUNT/ N_dy, N_hr_dy(1:24), N_mode_dy(1:4,1:4), N_GOOD, -
               N_mode_hr_dy(1:4,1:4,1:24), N_dy_hr(1:24), N_hr, -
               N_noLOS

* ***********************************************************************
* Initialize constants

      DATA P_index/1,2,3,4,5,6,4*0,7,8,0,9,10,0,11,3*0,-
                   12,0,13,0,14,0,15,3*0,16,17,18,19,20,5*0,-
                   21,0,22,23,24,5*0,25,26,27,17*0,28,29,30/

      DATA P_type/3*1,3,2*2,2*1,3,2,2,1,1,2,2,4*1,2,-
           1,1,3,2,3*1,3*1/

      DATA P_scale/10.0,1.0,100.0/

      DATA BLANK_line/'                '/

      DATA P_names/'foF2','fxF2','fzF2','M(3000)F2','h"F2','fpF2',-
      'foF1','fxF1','M(3000)F1','h"F1','h"F','foE','foE2','h"E','h"E2',-
      'foEs','fxEs','fbEs','fEs','h"Es','foF1.5','fmin','M(3000)F1.5',-
      'h"f1.5','foI','fxI','fmI','I(2000)','I','I(xxx)'/

      DATA MODE_name/'F2-F2','F1-F2',' E-F2','Es-F2', -
                   'F2-F1','F1-F1',' E-F1','Es-F1', -
```

```
     ' F2-E',' F1-E',' E-E',' Es-E', -
    'F2-Es','F1-Es',' E-Es','Es-Es'/

    DATA P_heading/' FRQ ',' HGT ',' '/
    DATA DIM_heading/'(MHz)',' (km)','FACTR','(deg)'/
    DATA HOUR_heading/' Universal Time in Hours '/
    DATA QUAL_heading/' QUAL'/
    DATA DESC_heading/' DESC'/
    DATA DIST_heading/'Path Length in Kilometers'/
    DATA L_heading/' FRQ ',' ANG ',' HGT '/
    DATA f_limit/2.0, 6.0, 10.0, 14.0, 18.0, -
          22.0, 26.0, 30.0, 40.0, 60.0, 100.0, 150.0, -
          6.0, 10.0, 14.0, 18.0, 22.0, 26.0, 30.0, -
          40.0, 60.0, 100.0, 150.0, 300.0/

    DATA N_DISP/12/

    DATA f_RANGE /' 2-6 ', ' 6-10 ', '10-14 ', '14-18 ', '18-22 ', -
          '22-26 ', '26-30 ', '30-40 ', '40-60 ', '60-100', -
         'to 150', 'to 300'/

    Rearth_km = 6370.0
    Re_km = 4.0 * Rearth_km / 3.0

C ***************************************************************************
C ***************************************************************************
C Open input file

    WRITE(6,*) 'Type name of input file with extension ".INP"'
    READ(6,600) file_INP

C Concatenate input and output file names

    INP_file = file_INP(1:LEN_TRIM(file_INP))//'.INP'
    OUT_file = file_INP(1:LEN_TRIM(file_INP))//'.OUT'
    LNK_file = file_INP(1:LEN_TRIM(file_INP))//'.LNK'

C Read input file

    OPEN(UNIT=2,STATUS='OLD',FILE=INP_file)
    READ(2,*) STATION_check(1), STATION_check(2)
    READ(2,*) HOUR_i, DAY_i, MONTH_i, YEAR_i
    READ(2,*) HOUR_f, DAY_f, MONTH_f, YEAR_f
    READ(2,*) ( YEAR_mask(j), j = 1, 25 )
    READ(2,*) ( MONTH_mask(j), j = 1, 12 )
    READ(2,*) MONTHLY_flag
    READ(2,*) ( DAY_mask(j), j = 1, 31 )
```

```
      READ(2,*) ( HOUR_mask(j), j = 1, 24 )
      READ(2,*) N_Dkm, ( D_km(j), j = 1, N_Dkm )
      READ(2,*) fMHz_low, fMHz_hgh, N_FRQ
      READ(2,*) dkm_low, dkm_hgh, N_DST
      READ(2,*) adg_low, adg_hgh, N_ANG
      DO i = 1, 30
          READ(2,*) j, CODE_mask(j), P_low(j), P_hgh(j), N_Pbins(j)
      END DO
      READ(2,*) DIV_flag
C     READ(2,*) N_FILE

C Read names of standard ionosonde files.  If an error occurs or if
C if no input filenames are present, output the appropriate message
C and stop execution

      i_FILE = 1

100   READ(2,*,ERR=998,END=999) S1_files(i_FILE), S2_files(i_FILE)

C Determine the number of ionosonde input files

      DO WHILE ( i_FILE .GT. 0 )
          i_FILE = i_FILE + 1
          READ(2,*,END=102) S1_files(i_FILE), S2_files(i_FILE)
      END DO

102   N_file = i_FILE - 1

C     i_FILE = 2
C     DO WHILE ( i_FILE .LE. N_FILE )
C       i_FILE = i_FILE + 1
C         READ(2,601) S1_files(i_FILE), S2_files(i_FILE)
C     END DO

      d_fMHz = ( fMHz_hgh - fMHz_low ) / FLOAT( N_FRQ )
      d_adg = ( adg_hgh - adg_low ) / FLOAT( N_ANG )
      d_dkm = ( dkm_hgh - dkm_low ) / FLOAT( N_DST )

C INITIALIZE all bin values to zero

      N_GOOD = 0.0
      month_old = 0
      adg_bin = 0.0
      adg_hr_bin = 0.0
      adg_mode_bin = 0.0
      adg_mode_hr_bin = 0.0
      AVG_adg_mode_hr = 0.0
```

```
      AVG_adg_mode = 0.0
      ANG_hr_frq = 0.0
      dkm_bin = 0.0
      dkm_hr_bin = 0.0
      dkm_mode_bin = 0.0
      dkm_mode_hr_bin = 0.0
      AVG_dkm_mode_hr = 0.0
      AVG_dkm_mode = 0.0
      fMHz_bin = 0.0
      fMHz_hr_bin = 0.0
      fMHz_mode_bin = 0.0
      fMHz_mode_hr_bin = 0.0
      AVG_fMHz_mode_hr = 0.0
      SDV_fMHz_mode_hr = 0.0
      AVG_fMHz_mode = 0.0
      SDV_fMHz_mode = 0.0
      fMHz_hr_disp = 0.0
      fMHz_disp = 0.0
      N_dy = 0.0
      N_hr = 0.0
      N_hr_dy = 0.0
      N_dy_hr = 0.0
      N_mode_dy = 0.0
      N_mode_hr_dy = 0.0
      N_months = 0
      N_CODE = 0

"  Initialize all ionosonde parameter storage locations to -1.0.  Note that
"  the commented parameters are not yet used in the calculation of forward
"  propagating signals.

      foF2 = -1.0
"      fxF2 = -1.0
"      fzF2 = -1.0
      M3000F2 = -1.0
      hF2 = -1.0
"      hpf2 = -1.0
      foF1 = -1.0
"      fxF1 = -1.0
      M3000F1 = -1.0
      hF1 = -1.0
      hF = -1.0
      foE = -1.0
"      foE2 = -1.0
      hE = -1.0
"      hE2 = -1.0
      foEs = -1.0
```

```
*     fxEs = -1.0
*     fbEs = -1.0
      fEs = -1.0
      hEs = -1.0
*     foF1d5 = -1.0
*     fmin = -1.0
*     M3000F1d5 = -1.0
*     hf1d5 = -1.0
*     foI = -1.0
*     fxI = -1.0
*     fmI = -1.0
*     I2000 = -1.0
*     Ixxx = -1.0

* ************************************************************************
* Print heading for OUTPUT file.
*
      IF ( MONTHLY_flag ) THEN
        OPEN(UNIT=8,STATUS='UNKNOWN',ACCESS='APPEND',FILE=LNK_file)
          WRITE(8,701) LNK_file
      ELSE
          OPEN(UNIT=3,STATUS='NEW',FILE=OUT_file)
      END IF

* START of FILE loop

      i_FILE = 0
      RUN_flag = .TRUE.
      GEOG_flag = .FALSE.
      DO WHILE ( i_FILE .LT. N_FILE .AND. RUN_flag )

* Initialize the FIRST_month flag to "TRUE", indicating that the
* first month of data in the current input files has not yet been (reread
* AND increment the input file counter to a "1".

          FIRST_month = .TRUE.
        i_FILE = i_FILE + 1

* Set both READ_flag values to "T" at the start of the parameter input loop

        READ_flag = .TRUE.

* Open input file for ionosonde data from current station

          OPEN(UNIT=4,STATUS='OLD',FILE=S1_files(i_FILE))
          OPEN(UNIT=7,STATUS='OLD',FILE=S2_files(i_FILE))
```

```
" Start of file-read loop.  "READ_flag" tells the program to keep
" reading parameter values for the current YEAR, MONTH, and DAY in the
" MONTH if set to "T".  If "F", then it discontinues reading the current
" station file.

      i_READ = 0
      ERR_LINE(1) = 0
      ERR_LINE(2) = 0

      DO WHILE ( READ_flag .AND. RUN_flag )

        i_READ = i_READ + 1

" Read the next line of the current input file and check the first
" 21 characters to make sure they are nonblank.  If they are blank,
" then a new record should be read.

      RECORD_I(1)(1:21) = BLANK_line
        DO WHILE( RECORD_I(1)(1:21) .EQ. BLANK_line )
          ERR_LINE(1) = ERR_LINE(1) + 1
          READ(4,702,END=200) RECORD_I(1)
        END DO

      RECORD_I(2)(1:21) = BLANK_line
      DO WHILE( RECORD_I(2)(1:21) .EQ. BLANK_line )
          ERR_LINE(2) = ERR_LINE(2) + 1
          READ(7,702,END=200) RECORD_I(2)
        END DO

" Extract station name, code, standard time meridian, geographic co-latitude,
" east longitude, year, month, and the parameter code

        DO i_SITE = 1, 2

          STATION_name(i_SITE) = RECORD_I(i_SITE)(1:20)

          IF ( STATION_name(i_SITE) .NE. -
               STATION_check(i_SITE) ) THEN
            WRITE(6,*)
            WRITE(6,*) 'ERR:  Station name is not input name!'
            WRITE(6,*) '1', '  STATION_check:  ', STATION_check(1)
            WRITE(6,*) 'STATION_name:  ', STATION_name(1)
            WRITE(6,*) 'FILE 1 LINE: ', ERR_LINE(1)
            WRITE(6,*)
            WRITE(6,*) '2', '  STATION_check:  ', STATION_check(2)
            WRITE(6,*) 'STATION_name:  ', STATION_name(2)
            WRITE(6,*) 'FILE 2 LINE: ', ERR_LINE(2)
```

```
          WRITE(6,*) '     Execution has been halted.'
        WRITE(6,*) ' Please check station names and try again.'
        STOP
      END IF

      STATION_code(i_SITE) = RECORD_I(i_SITE)(21:26)
      TIME_meridian(i_SITE) = RECORD_I(i_SITE)(26:29)
      STATION_COLAT_ddg(i_SITE) = RECORD_I(i_SITE)(30:33)
      STATION_ELNG_ddg(i_SITE) = RECORD_I(i_SITE)(34:37)

      END DO

* Extract the YEAR from the current input file.  If the YEAR is not to
* be included in the analysis (YEAR_mask) or it falls outside of the
* desired processing interval, then exit loop which reads file for current
* YEAR for both sites, i.e., do not read the corresponding file for the
* second site.

      DO i_SITE = 1, 2

        YEAR(i_SITE) = RECORD_I(i_SITE)(38:41)
        Y_1 = ICHAR( RECORD_I(i_SITE)(38:38) ) - 48
        Y_2 = ICHAR( RECORD_I(i_SITE)(39:39) ) - 48
        Y_3 = ICHAR( RECORD_I(i_SITE)(40:40) ) - 48
        Y_4 = ICHAR( RECORD_I(i_SITE)(41:41) ) - 48
          i_YEAR = 1000 * Y_1 + 100 * Y_2 + 10 * Y_3 + Y_4
          IF ( i_YEAR .LT. YEAR_i .OR. i_YEAR .GT. YEAR_f .OR.-
           .NOT. YEAR_mask(i_FILE) ) THEN
          READ_flag = .FALSE.
        END IF

        END DO

        IF ( .NOT. READ_flag ) CYCLE

        IF ( YEAR(1) .NE. YEAR(2) ) THEN
        WRITE(6,*) 'ERR:  The YEAR values differ between the two'
        WRITE(6,*) '      ionosonde input files!'
        WRITE(6,*) 'ACT:  Execution has been halted.'
        WRITE(6,*) 'Please check these files and try again.'
        STOP
      END IF

* Extract the MONTH from the current input file.  If the MONTH is not to
* be included inthe analysis (MONTH_mask) or it falls outside of the
* desired processing interval, then exit loop which reads file for current
* MONTH for both sites, i.e., do not read the corresponding file for the
```

```
" second site.  If this is the first input file, initialize MONTH_1 to
" i_MONTH.

      DO i_SITE = 1, 2
        MONTH(i_SITE) = RECORD_I(i_SITE)(42:43)
        M_1 = ICHAR( RECORD_I(i_SITE)(42:42) ) - 48
        M_2 = ICHAR( RECORD_I(i_SITE)(43:43) ) - 48
        i_MONTH = 10 * M_1 + M_2
        END DO

        IF ( MONTH(1) .NE. MONTH(2) ) THEN
        WRITE(6,*) 'ERR:  The MONTH values differ between the two'
        WRITE(6,*) '      ionosonde input files!'
        WRITE(6,*) 'ACT:  Execution has been halted.'
        WRITE(6,*) 'Please check these files and try again.'
        STOP
      END IF

" If the current YEAR is the final year, check to see if the final MONTH
" has been exceeded.  If so, stop reading parameter blocks for the current
" SITE and either look at the next SITE's file or exit to process the
" results.

      IF ( i_YEAR .EQ. YEAR_f .AND. i_MONTH .GT. MONTH_f ) THEN
        READ_flag = .FALSE.
        CYCLE
      END IF

" If the current MONTH data block is not to be processed or if the first
" MONTH data block to be processed has not yet been encountered, then read
" the next 31 records to skip the current data block and read the next
" data block.

      IF ( .NOT. MONTH_mask( i_MONTH ) .OR. -
         ( i_YEAR .EQ. YEAR_i .AND. i_MONTH .LT. MONTH_i ) ) THEN
        DO i = 1, 31
            ERR_LINE(1) = ERR_LINE(1) + 1
            READ(4,702,END=200) RECORD_I(1)
            ERR_LINE(2) = ERR_LINE(2) + 1
            READ(7,702,END=200) RECORD_I(2)
          END DO
        CYCLE
      END IF

        IF ( FIRST_month ) THEN
          MONTH_1 = i_MONTH
        FIRST_month = .FALSE.
```

```
      END IF

* If this is the first file read, determine the latitude and longitude
* of the current ionosonde station site and then compute the inter-site
* Great Circle path length and bearing angles.

      IF ( .NOT. GEOG_flag ) THEN

        DO i_SITE = 1, 2

        L_1 = ICHAR( RECORD_I(i_SITE)(30:30) ) - 48
        L_2 = ICHAR( RECORD_I(i_SITE)(31:31) ) - 48
        L_3 = ICHAR( RECORD_I(i_SITE)(32:32) ) - 48
        L_4 = ICHAR( RECORD_I(i_SITE)(33:33) ) - 48
        I_DUM = 100 * L_1 + 10 * L_2 + L_3
        COLAT_dg(i_SITE) = FLOAT( I_DUM ) + 0.1 * FLOAT( L_4 )

        L_1 = ICHAR( RECORD_I(i_SITE)(34:34) ) - 48
        L_2 = ICHAR( RECORD_I(i_SITE)(35:35) ) - 48
        L_3 = ICHAR( RECORD_I(i_SITE)(36:36) ) - 48
        L_4 = ICHAR( RECORD_I(i_SITE)(37:37) ) - 48
        I_DUM = 100 * L_1 + 10 * L_2 + L_3
        LNG_dg(i_SITE) = FLOAT( I_DUM ) + 0.1 * FLOAT( L_4 )

        END DO

       CALL BEARING( STATION_name, COLAT_dg, LNG_dg, Ds_km, -
            As_rd, azimuth_dg, MONTHLY_flag )

        GEOG_flag = .TRUE.

      END IF

* Read and decode the parameter code whose values are contained in the
* next-to-be-read input block for the current file and ionosonde station.

      DO i_SITE = 1, 2
         P_code(1) = ICHAR( RECORD_I(i_SITE)(44:44) ) - 48
         P_code(2) = ICHAR( RECORD_I(i_SITE)(45:45) ) - 48
         PARM_code(i_SITE) = 10 * P_code(1) + P_code(2)
       j_CODE(i_SITE) = P_index( PARM_code(i_SITE) + 1 )
      END DO

      IF ( j_CODE(1) .NE. j_CODE(2) ) THEN
        WRITE(6,*) 'ERR: The PARMAMETER values differ between'
        WRITE(6,*) '     the two ionosonde input files!'
        WRITE(6,*) 'ACT: Execution has been halted.'
```

```
          WRITE(6,*) 'Please check these files and try again.'
          STOP
        ELSE
          i_CODE = j_CODE(1)
          END IF

          WRITE(6,*) 'MONTH: ', i_MONTH,'  PARM: ', i_CODE

"  If not all of the parameters have yet been read from the current file
"  AND the current parameter code exceeds the code counter "N_CODE", then
"  set "N_CODE" equal to the parameter CODE counter "i_CODE". This state
"  implies that not all of a month's data blocks have yet been input and
"  processing of the monthly data cannot proceed.
"  Otherwise, determine if all of the data blocks necessary to process
"  the current month's data have been read for both stations.  If so,
"  process the month's data.

        IF ( i_MONTH .EQ. MONTH_1 ) THEN

          IF ( i_CODE .LE. N_CODE ) THEN

"  "i_CODE < N_CODE" means that N_CODE has been reached a maximum from
"  a first pass through all blocks in a previous month's-worth of data.
"  In this case, the first month's data has been read and the first block
"  in the second month's blocks has been detected.  Process the first months
"  data
            DO i_DAY = 1, 31
              DO i_HOUR = 1, 24
                    CALL HFLINK2( i_DAY, i_HOUR, LINK_flag, -
                      MONTHLY_flag )
              END DO
            END DO

              IF ( month_OLD .EQ. 0 ) month_OLD = i_MONTH - 1
            IF ( month_OLD .EQ. 0 ) month_OLD = 12

              IF ( MONTHLY_flag ) THEN
              CALL NRMLYZE( i_YEAR, month_OLD, MONTHLY_flag )
            END IF

            N_months = N_months + 1
            month_OLD = i_MONTH
              FIRST_month = .FALSE.
              OLD_MONTH = MONTH(1)

            ELSE
```

```
            N_CODE = i_CODE

          END IF

        ELSE IF ( i_MONTH .NE. MONTH_1 ) THEN

            DO i_DAY = 1, 31
              DO i_HOUR = 1, 24
                    CALL HFLINK2( i_DAY, i_HOUR, LINK_flag, -
                      MONTHLY_flag )
              END DO
            END DO

            IF ( month_OLD .EQ. 0 ) month_OLD = i_MONTH - 1
            IF ( month_OLD .EQ. 0 ) month_OLD = 12

                IF ( MONTHLY_flag ) THEN
              CALL NRMLYZE( i_YEAR, month_OLD, MONTHLY_flag )
            END IF

            N_months = N_months + 1
            month_OLD = i_MONTH
                FIRST_month = .FALSE.
                OLD_MONTH = MONTH(1)

          END IF

* If the current parameter code is not to be used, read the next data
* block and return to start of loop.

        CODE_flag = .TRUE.

          IF ( .NOT. CODE_mask( i_CODE ) ) THEN
            DO i = 1, 31
                ERR_LINE(1) = ERR_LINE(1) + 1
                READ(4,702,END=200) RECORD_I(1)
                ERR_LINE(2) = ERR_LINE(2) + 1
                READ(7,702,END=200) RECORD_I(2)
            END DO
            CODE_flag = .FALSE.
          END IF

          IF ( .NOT. CODE_flag ) CYCLE

* Determine the upper and lower bin limits of the parameter for either
* frequency (i_DIM = 1) or height (i_DIM = 2)
```

```
      i_DIM = P_type( i_CODE )
      N_bins = N_Pbins( i_DIM )

"  Compute parameter increment value

      d_P = ( P_hgh(i_CODE) - P_low(i_CODE) ) -
          / FLOAT( N_Pbins(i_CODE) )

"  START loop for each day of the current month

      i_DAY = 0
      DAY_flag = .TRUE.

      DO WHILE ( i_DAY .LT. 31 .AND. DAY_flag )

        i_DAY = i_DAY + 1

"  Read next input record and divide into 24 5-character strings
"  Check the first 20 characters to make sure they are nonblank.  If they
"  are blank, then a new record should be read.

          ERR_LINE(1) = ERR_LINE(1) + 1
          READ(4,702,END=997) RECORD_I(1)
          ERR_LINE(2) = ERR_LINE(2) + 1
        READ(7,702,END=997) RECORD_I(2)

"         WRITE(6,*) 'SITE 1 INPUT RECORD'
"         WRITE(6,*) RECORD_I(1)
"         WRITE(6,*) 'SITE 2 INPUT RECORD'
"         WRITE(6,*) RECORD_I(2)

"  Check to see if current day is within interval to be processed.

        IF ( ( i_YEAR .EQ. YEAR_i .AND. i_MONTH .EQ. MONTH_i -
           .AND. i_DAY .LT. DAY_i ) .OR. -
           .NOT. DAY_mask(i_DAY) ) CYCLE

        IF ( i_YEAR .EQ. YEAR_f .AND. i_MONTH .EQ. MONTH_f -
           .AND. i_DAY .GT. DAY_f ) THEN
          DAY_flag = .FALSE.
          READ_flag = .FALSE.
          CYCLE
        END IF

"  START loop for each hour of the current day.

        HOUR_flag = .TRUE.
```

```
      i_HOUR = 0
      DO WHILE ( i_HOUR .LT. 24 .AND. HOUR_flag )

        i_HOUR = i_HOUR + 1

        N_dy_hr(i_HOUR) = N_dy_hr(i_HOUR) + 1.0
        N_hr = N_hr + 1.0

*     Check to see if current hour is within interval to be processed

*           IF ( ( i_YEAR .EQ. YEAR_i .AND. i_MONTH .EQ. MONTH_i -
*                .AND. i_DAY .EQ. DAY_i .AND. i_HOUR .LT. HOUR_i ) -
*              .OR. .NOT. HOUR_mask(i_HOUR) ) CYCLE
*
*           IF ( i_YEAR .EQ. YEAR_f .AND. i_MONTH .EQ. MONTH_f -
*             .AND. i_DAY .EQ. DAY_f .AND. i_HOUR .GT. HOUR_f ) THEN
*             HOUR_flag = .FALSE.
*             DAY_flag = .FALSE.
*             Rept_flag = .FALSE.
*             CYCLE
*           END IF

*     Determine the parameter value for each ionosonde station.

        k_SITE = 0

        DO i_SITE = 1, 2

          P_value = 0.0

          k_CHAR = 5 * ( i_HOUR - 1 ) + 1

          IF ( RECORD_I(i_SITE)(k_CHAR:k_CHAR+4) -
             .EQ. '     ' ) CYCLE

*     Read scaled parameter for the current hour at the ionosonde.

          P_char(1) = RECORD_I(i_SITE)(k_CHAR:k_CHAR)
          P_char(2) = RECORD_I(i_SITE)(k_CHAR+1:k_CHAR+1)
          P_char(3) = RECORD_I(i_SITE)(k_CHAR+2:k_CHAR+2)
          Q_char = RECORD_I(i_SITE)(k_CHAR+3:k_CHAR+3)
          D_char = RECORD_I(i_SITE)(k_CHAR+4:k_CHAR+4)

          IF ( P_char(1) .NE. ' ' ) THEN

            P_1 = REAL( ICHAR( P_char(1) ) - 48 )
            P_2 = REAL( ICHAR( P_char(2) ) - 48 )
```

```
            P_3 = REAL( ICHAR( P_char(3) ) - 48 )
            P_flag = .TRUE.

          ELSE IF ( P_char(2) .NE. ' ' ) THEN

            P_1 = 0.0
            P_2 = REAL( ICHAR( P_char(2) ) - 48 )
            P_3 = REAL( ICHAR( P_char(3) ) - 48 )
            P_flag = .TRUE.

          ELSE IF ( P_char(3) .NE. ' ' ) THEN

              P_1 = 0.0
              P_2 = 0.0
              P_3 = REAL( ICHAR( P_char(3) ) - 48 )
              P_flag = .TRUE.

          ELSE

            P_value = 0.0
            P_flag = .FALSE.

          END IF

          IF ( P_flag ) THEN

            P_value = ( 100.0*P_1 + 10.0*P_2 + P_3 ) -
                  / P_scale(i_DIM)

            k_SITE = k_SITE + 1

            CALL PARMBN2( i_SITE, i_DAY, i_HOUR, i_CODE, -
                  P_value, Q_char, D_char )

          END IF

" End of station loop.

          IF ( k_SITE .EQ. 2 ) N_GOOD = N_GOOD + 1.0

        END DO

" End of hourly loop for current day, parameter, month, and year (file).

      END DO

" End of daily loop for current parameter, month, and year (file).
```

```
          END DO

*  End of current parameter & month loop for year (file).

          END DO

*  All input records in the current input file (or year) have been read.

200       CLOS'       IT=4)

*  Input results from next year (or file)

      END DO

*  Compute HF links for last month

      DO i_DAY = 1, 31
          DO i_HOUR = 1, 24
            CALL HFLINK2( i_DAY, i_HOUR, LINK_flag, MONTHLY_flag )
        END DO
      END DO

      N_months = N_months + 1

      IF ( MONTHLY_flag ) THEN
          CALL NRMLYZE( i_YEAR, month_OLD, MONTHLY_flag )
      END IF

      month_OLD = i_MONTH
      FIRST_month = .FALSE.

*  Output the probability distributions for each parameter code, both
*  for each hour and over all hours in a column format

      IF ( .NOT. MONTHLY_flag ) THEN

*  All input files have been read.  Normalize each distribution count
*  by the numbers of samples for each paramater code.

          CALL NRMLYZE( i_YEAR, i_MONTH, MONTHLY_flag )

          WRITE(3,704) STATION_name(1), STATION_name(2)
704       FORMAT(/10x,'IONOSONDE PARAMETER STATISTICS AT ', -
                  /10x, a20, ' & ', a20 )

        WRITE(3,705) HOUR_i, DAY_i, MONTH_i, YEAR_i, HOUR_f, DAY_f,-
```

```
            MONTH_f, YEAR_f
705     FORMAT(/10x,'IONOSONDE measurements processed for the period: ', -
            /10x, 'From: ', 3(i2,', '),i4,' to ',3(i2,', '),i4)

"  Output HF link propagation characteristics.

        WRITE(3,730)
730     FORMAT(//10x,'HF LINK PROPAGATION VERSUS HOUR & 2-HOP MODE')

"  Output hourly results for each two-hop propagation mode

        DO i_MODE = 1, 4

          DO j_MODE = 1, 4

            WRITE(3,734) 'PROPAGATION MODE: ', -
                    MODE_name(i_MODE,j_MODE), -
                    ( i, i = 1, 24 )
            WRITE(3,736) '100 Pr', -
                (100.0*N_mode_hr_dy(i_MODE,j_MODE,i_HOUR),i_HOUR=1,24), -
                      100.0*N_mode_dy(i_MODE,j_MODE)
            WRITE(3,737) 'AVGFRQ', -
                (AVG_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR), -
                i_HOUR=1,24), AVG_fMHz_mode(i_MODE,j_MODE)
            WRITE(3,737) 'SDVFRQ', -
                (SDV_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR), -
                i_HOUR=1,24), SDV_fMHz_mode(i_MODE,j_MODE)
"           WRITE(3,737) 'AVGANG', -
"                (AVG_adg_mode_hr(i_MODE,j_MODE,i_HOUR), -
"                i_HOUR=1,24), AVG_adg_mode(i_MODE,j_MODE)
"           WRITE(3,737) 'AVGDST', -
"                (AVG_dkm_mode_hr(i_MODE,j_MODE,i_HOUR), -
"                i_HOUR=1,24), AVG_dkm_mode(i_MODE,j_MODE)

          END DO

        END DO

        WRITE(3,732)
        WRITE(3,738) 2.0*Ds_km, ' FREQ ', '(MHZ) ', ( i, i = 1, 24 )

        DO i_DISP = 1, N_DISP

          WRITE(3,742) f_RANGE(i_DISP), -
              ( 100.0*fMHz_hr_disp(i_HOUR,i_DISP), i_HOUR = 1 , 24 ), -
                    100*fMHz_disp(i_DISP)
```

```
        END DO

      WRITE(6,*) '"GOOD" hour percentage: ', N_GOOD / N_hr
      WRITE(6,*) 'Percentage of NO LOS hours: ', N_noLOS / N_hr

*     WRITE(3,746)
*     WRITE(3,748) 2.0*Ds_km, 'ANGLE ', ( i, i = 1, 24 )
*     DO i_ANG = 0, N_ANG
*
*       a_dg = adg_low + i_ANG * d_a
*
*       WRITE(3,752) a_dg,(adg_hr_bin(i_HOUR,i_ANG),i_HOUR=1,24), -
*                    adg_bin(i_ANG)
*
*     END DO
*     WRITE(3,754) (adg_hr_bin(i_HOUR,N_ANG+1),i_HOUR=1,24), -
*                  adg_bin(N_ANG+1)
*
*     WRITE(3,756)
*     WRITE(3,758) 2.0*Ds_km, ' RANGE', ( i, i = 1, 24 )
*
*     DO i_DST = 0, N_DST
*
*      dkm = dkm_low + i_HGT * d_dkm
*
*        WRITE(3,762) dkm, -
*             (dkm_hr_bin(i_HOUR,i_DST),i_HOUR=1,24), -
*             dkm_bin(i_DST)
*
*     END DO
*    WRITE(3,764) (dkm_hr_bin(i_HOUR,N_DST+1),i_HOUR=1,24), -
*           dkm_bin(N_DST+1)
*
* End of distance loop

    END IF

    CLOSE(UNIT=2)

    IF ( MONTHLY_flag ) THEN
      CLOSE(UNIT=8)
    ELSE
      CLOSE(UNIT=3)
    END IF

    STOP
```

```
997  WRITE(6,*) 'Error on input, probably wrong number of -
                 characters in record'
     CLOSE(UNIT=2)
     CLOSE(UNIT=3)
     STOP

998  WRITE(6,*) 'Error on input, probably no access to input file'
     CLOSE(UNIT=2)
     CLOSE(UNIT=3)
     STOP

999  WRITE(6,*) 'No ionosonde input files were listed'
     CLOSE(UNIT=2)
     CLOSE(UNIT=3)
     STOP

600  FORMAT(a20)
601  FORMAT(a25,1x,a25)
700  FORMAT(a20,1x,a20)
701  FORMAT(1x,'Input filename: ',a30)
702  FORMAT(a120)
734  FORMAT(//1x, a19, a5, 33x, 'Hour of the day' -
          / 7x, 24( 1x, i3, 1x ), ' NET ' )
736  FORMAT( 1x, a6, 25(f4.1,1x) )
737  FORMAT( 1x, a6, 25(f4.1,1x) )
732  FORMAT(//10x,'***** MAXIMUM USABLE FREQUENCY *****')
738  FORMAT(//5x,'          Link Range: ', f7.1, ' KM' -
          //1x, a6, 48x,'Hour of the day (UT)' -
          /1x, a5, 24( 2x, i3 ), ' NET ' )
742  FORMAT( 1x, a6, 25(f4.1,1x) )
746  FORMAT('1'//10x,'***** RAY ELEVATION ANGLE *****')
748  FORMAT(//5x, '          Link Range: ', f7.1, ' KM' -
          //1x, a6, 48x, 'Hour of the day' -
          /7x, 24( i3, 2x ), ' NET ' )
752  FORMAT( 1x, f6.2, 25f5.3 )
754  FORMAT( 1x, '  > ', 25f5.3 )
756  FORMAT('1'//10x,'***** FIRST-HOP PATH LENGTH *****')
764  FORMAT(1x,'  > ', 25f5.3 )
758  FORMAT(//5x, '          Link Range: ', f7.1, ' KM' -
        //1x, a6, 48x, 'Hour of the day' -
        /7x, 24( i3, 2x ), ' NET ' )
762  FORMAT( 1x, f6.1, 25f5.3 )

     END
```

```
      SUBROUTINE NRMLYZE( i_YEAR, i_MONTH, MONTHLY_flag )

*     Declare arrays for HF link calculations

      REAL*4 fMHz_low, fMHz_hgh, dkm_low, dkm_hgh
      REAL*4 fMHz_hr_disp, fMHz_disp
      REAL*4 adg_low, adg_hgh, d_fMHz, d_dkm
      REAL*4 a_limit, adg_bin, adg_hr_bin, adg_mode_bin
      REAL*4 adg_mode_hr_bin, AVG_adg_mode_hr, AVG_adg_mode
      REAL*4 d_limit, dkm_bin, dkm_hr_bin, dkm_mode_bin
      REAL*4 dkm_mode_hr_bin, AVG_dkm_mode_hr, AVG_dkm_mode
      REAL*4 f_limit, fMHz_bin, fMHz_hr_bin, fMHz_mode_bin
      REAL*4 fMHz_mode_hr_bin, AVG_fMHz_mode_hr, AVG_fMHz_mode
      REAL*4 SDV_fMHz_mode_hr, SDV_fMHz_mode, N_dy_hr, N_hr, N_GOOD
      REAL*4 N_dy, N_hr_dy, N_mode_dy, N_mode_hr_dy, N_smp, N_noLOS

      INTEGER*4 i_DAY, i_HOUR, i_DST, i_FRQ, i_MODE, i_SITE
      INTEGER*4 N_DST, N_FRQ, N_ANG
      INTEGER*4 N_bins, month_OLD

      CHARACTER*5 L_heading(3), MODE_name(1:4,1:4)
      CHARACTER*25 DIST_heading

*     Declare variables for MUF calculations

      REAL*4 m_MUF, b_MUF, DUM_1, DUM_2, N_km_hr, N_km
      REAL*4 MUF_10(1:24), MUF_50(1:24), MUF_90(1:24)
      REAL*4 ANG_10(1:24), ANG_50(1:24), ANG_90(1:24)

      INTEGER*4 i_YEAR, i_MONTH, N_DISP

      LOGICAL*1 P10_FLAG, P50_FLAG, P90_FLAG, MONTHLY_flag, NONZERO_flag

      LOGICAL*1 TEST_FLAG

      CHARACTER*25 LNK_file

*     Define all common blocks.

      COMMON/IONSNP/ foF2(2,31,24), M3000F2(2,31,24), hF2(2,31,24), -
                     foF1(2,31,24), M3000F1(2,31,24), hF1(2,31,24), -
                     hF(2,31,24), foE(2,31,24), -
                     hE(2,31,24), foEs(2,31,24), fbEs(2,31,24), -
                         fEs(2,31,24), hEs(2,31,24)

*                        fmin(2,31,24), fxI(2,31,24)
```

```
      COMMON /MISC/ Re_km, Rearth_km

      COMMON /LIMIT/ fMHz_low, fMHz_hgh, N_FRQ, dkm_low, dkm_hgh, N_DST, -
     &        adg_low, adg_hgh, N_ANG, LNK_file, d_fMHz, d_dkm, -
     &        d_adg, As_rd, Ds_km

*     COMMON /ANGLE/ adg_bin(0:61), -
*    &        adg_hr_bin(1:24,0:61), adg_mode_bin(1:4,1:4,0:61), -
*    &        adg_mode_hr_bin(1:4,1:4,1:24,0:61), -
*    &        AVG_adg_mode_hr(1:4,1:4,1:24), AVG_adg_mode(1:4,1:4), -
*    &        ANG_hr_frq(1:24,0:61,1:2)

      COMMON /ANGLE/ ANG_hr_frq(1:24,0:61,1:2)

*     COMMON /RANGE/ dkm_bin(0:61), -
*    &        dkm_hr_bin(1:24,0:61), dkm_mode_bin(1:4,1:4,0:61), -
*    &        dkm_mode_hr_bin(1:4,1:4,1:24,0:61), -
*    &        AVG_dkm_mode_hr(1:4,1:4,1:24), AVG_dkm_mode(1:4,1:4)

      COMMON /FRQBN/ f_limit(1:12,1:2), fMHz_bin(0:61), -
     &        fMHz_hr_bin(1:24,0:61), fMHz_mode_bin(1:4,1:4,0:61), -
     &        fMHz_mode_hr_bin(1:4,1:4,1:24,0:61), -
     &        AVG_fMHz_mode_hr(1:4,1:4,1:24), -
     &        SDV_fMHz_mode_hr(1:4,1:4,1:24), -
     &            AVG_fMHz_mode(1:4,1:4), SDV_fMHz_mode(1:4,1:4), -
     &            fMHz_hr_disp(1:24,1:12), fMHz_disp(1:12)

      COMMON /COUNT/ N_dy, N_hr_dy(1:24), N_mode_dy(1:4,1:4), N_GOOD, -
     &        N_mode_hr_dy(1:4,1:4,1:24), N_dy_hr(1:24), N_hr, -
     &        N_noLOS

      DATA N_DISP/12/

      IF ( N_dy .LT. 0.5 ) RETURN

      DO i_FRQ = 0, N_FRQ+1
          fMHz_bin(i_FRQ) = fMHz_bin(i_FRQ) / N_dy
      END DO

*     DO i_ANG = 1, N_ANG
*       adg_bin(i_ANG) = adg_bin(i_ANG) / N_dy
*     END DO
*
*     DO i_DST = 1, N_DST
*       dkm_bin(i_DST) = dkm_bin(i_DST) / N_dy
*     END DO
```

```
      DO i_DISP = 1, N_DISP
        fMHz_disp(i_DISP) = fMHz_disp(i_DISP) / N_hr
      END DO

      DO i_HOUR = 1, 24

        N_smp = N_dy_hr(i_HOUR)

          IF ( N_smp .LT. 0.5 ) CYCLE

          DO i_DISP = 1, N_DISP
            fMHz_hr_disp(i_HOUR,i_DISP) = fMHz_hr_disp(i_HOUR,i_DISP) / N_smp
          END DO

        N_smp = N_hr_dy(i_HOUR)

        IF ( N_smp .LT. 0.5 ) CYCLE

        DO i_FRQ = 0, N_FRQ+1
            fMHz_hr_bin(i_HOUR,i_FRQ) = fMHz_hr_bin(i_HOUR,i_FRQ) / N_smp
          IF ( i_FRQ .GT. 0 .AND. MONTHLY_flag ) THEN
            fMHz_hr_bin(i_HOUR,i_FRQ) = fMHz_hr_bin(i_HOUR,i_FRQ) -
                              + fMHz_hr_bin(i_HOUR,i_FRQ-1)
            END IF
          END DO

"       DO i_ANG = 1, N_ANG
"         adg_hr_bin(i_HOUR,i_ANG) = adg_hr_bin(i_HOUR,i_ANG) -
"                           / N_smp
"       END DO
"
"       DO i_DST = 1, N_DST
"         dkm_hr_bin(i_HOUR,i_DST) = dkm_hr_bin(i_HOUR,i_DST) -
"                                 / N_smp
"       END DO

      DO i_MODE = 1, 4

        DO j_MODE = 1, 4

" Compute the PDFs for each bin, mode, and hour

            N_smp = N_mode_hr_dy(i_MODE,j_MODE,i_HOUR)
            IF ( N_smp .LT. 0.5 ) CYCLE

            N_mode_hr_dy(i_MODE,j_MODE,i_HOUR) = -
                    N_mode_hr_dy(i_MODE,j_MODE,i_HOUR) / N_dy
```

```
            DO i_FRQ = 1, N_FRQ
            fMHz_mode_hr_bin(i_MODE,j_MODE,i_HOUR,i_FRQ) = -
                  fMHz_mode_hr_bin(i_MODE,j_MODE,i_HOUR,i_FRQ) -
                        / N_smp
          END DO

"         DO i_ANG = 1, N_ANG
"           adg_mode_hr_bin(i_MODE,j_MODE,i_HOUR,i_ANG) = -
"                 adg_mode_hr_bin(i_MODE,j_MODE,i_HOUR,i_ANG) -
"                 / N_smp
"         END DO
"
"         DO i_DST = 1, N_DST
"           dkm_mode_hr_bin(i_MODE,j_MODE,i_HOUR,i_DST) = -
"                 dkm_mode_hr_bin(i_MODE,j_MODE,i_HOUR,i_DST) -
"                 / N_smp
"         END DO

" Compute the statistics for each mode

          SUM_fMHz = AVG_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR)
            SDV2_fMHz = SDV_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR)

          AVG_fMHz = SUM_fMHz / N_smp
          AVG_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR) = AVG_fMHz

            IF ( N_smp .GT. 1.5 ) THEN
                SDV_fMHz = N_smp * SDV2_fMHz - SUM_fMHz**2
                IF ( SDV_fMHz .LT. 0.0 ) SDV_fMHz = 0.0
                SDV_fMHz = SQRT( SDV_fMHz / ( N_smp * ( N_smp - 1.0 ) ) )
                SDV_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR) = SDV_fMHz
            END IF

"         SUM_adg = AVG_adg_mode_hr(i_MODE,j_MODE,i_HOUR)
"         AVG_adg = SUM_adg / N_smp
"         AVG_adg_mode_hr(i_MODE,j_MODE,i_HOUR) = AVG_adg
"
"         SUM_dkm = AVG_dkm_mode_hr(i_MODE,j_MODE,i_HOUR)
"         AVG_dkm = SUM_dkm / N_smp
"         AVG_dkm_mode_hr(i_MODE,j_MODE,i_HOUR) = AVG_dkm

        END DO

      END DO

    END DO
```

```
      DO i_MODE = 1, 4

        DO j_MODE = 1, 4

          N_smp = N_mode_dy(i_MODE,j_MODE)
          IF ( N_smp .LT. 0.5 ) CYCLE

          N_mode_dy(i_MODE,j_MODE) = -
                              N_mode_dy(i_MODE,j_MODE) / N_dy

          SUM_fMHz = AVG_fMHz_mode(i_MODE,j_MODE)
          SDV2_fMHz = SDV_fMHz_mode(i_MODE,j_MODE)

          AVG_fMHz = SUM_fMHz / N_smp
          AVG_fMHz_mode(i_MODE,j_MODE) = AVG_fMHz

          IF ( N_smp .GT. 1.5 ) THEN
            SDV_fMHz = n_smp * SDV2_fMHz - SUM_fMHz**2
            IF ( SDV_fMHz .LT. 0.0 ) SDV_fMHz = 0.0
            SDV_fMHz = SQRT( SDV_fMHz / ( N_smp * ( N_smp - 1.0 ) ) )
            SDV_fMHz_mode(i_MODE,j_MODE) = SDV_fMHz
          END IF

*         SUM_adg = AVG_adg_mode(i_MODE,j_MODE)
*         AVG_adg = SUM_adg / N_smp
*         AVG_adg_mode(i_MODE,j_MODE) = AVG_adg
*
*         SUM_dkm = AVG_dkm_mode(i_MODE,j_MODE)
*         AVG_dkm = SUM_dkm / N_smp
*         AVG_dkm_mode(i_MODE,j_MODE) = AVG_dkm
*
          DO i_FRQ = 1, N_FRQ
            fMHz_mode_bin(i_MODE,j_MODE,i_FRQ) = -
                              fMHz_mode_bin(i_MODE,j_MODE,i_FRQ) / N_smp
          END DO

*         DO i_ANG = 1, N_ANG
*           adg_mode_bin(i_MODE,j_MODE,i_ANG) = -
*                     adg_mode_bin(i_MODE,j_MODE,i_ANG) / N_smp
*         END DO
*
*         DO i_DST = 1, N_DST
*           dkm_mode_bin(i_MODE,j_MODE,i_DST) = -
*                     dkm_mode_bin(i_MODE,j_MODE,i_DST) / N_smp
*         END DO
```

```
      END DO

    END DO

" Exit if statistics are not being collected monthly.

" Output percentage of "good" hours in which a usable path was found.

    IF ( .NOT. MONTHLY_flag ) RETURN

" Compute tenth, fiftieth, and ninetieth percentiles for the link MUF
" add results to the LNK_file

    DO i_HOUR = 1, 24

      TEST_FLAG = .FALSE.
      IF ( i_HOUR .EQ. 10 .AND. i_MONTH .EQ. 2 ) TEST_FLAG = .TRUE.

      DO i_FRQ = 0, N_FRQ+1

      DUM_2 = fMHz_hr_bin(i_HOUR,i_FRQ)
      DUM_1 = fMHz_hr_bin(i_HOUR,N_FRQ+1)

      IF ( DUM_1 .GT. 0.0 ) THEN
        fMHz_hr_bin(i_HOUR,i_FRQ) = DUM_2 / DUM_1
      ELSE
        fMHz_hr_bin(i_HOUR,i_FRQ) = 0.0
      END IF

      f = fMHz_low + FLOAT( i_FRQ-1) * d_fMHz + 0.5 * d_fMHz

      END DO

     P10_FLAG = .FALSE.
     P50_FLAG = .FALSE.
       P90_FLAG = .FALSE.

      DO i_FRQ = 1, N_FRQ-1

        x2 = fMHz_hr_bin(i_HOUR,i_FRQ+1)

     IF ( x2 .GT. 0.10 .AND. .NOT. P10_FLAG ) THEN

       P10_FLAG = .TRUE.

       x1 = fMHz_hr_bin(i_HOUR,i_FRQ)
         y1 = fMHz_low + ( FLOAT( i_FRQ - 1 ) + 0.5 ) * d_fMHz
```

```
      y2 = y1 + d_fMHz
      DUM_1 = x2 - x1

    IF ( DUM_1 .NE. 0.0 ) THEN
      m_MUF = ( y2 - y1 ) / DUM_1
      b_MUF = ( y1 * x2 - y2 * x1 ) / DUM_1
      DUM_2 = m_MUF * 0.10 + b_MUF
    ELSE
          DUM_2 = fMHz_low + ( FLOAT( i_FRQ - 1 ) + 0.5 ) * d_fMHz
    END IF

    IF ( DUM_2 .GE. fMHz_low ) THEN
      MUF_10(i_HOUR) = DUM_2
    ELSE
          MUF_10(i_HOUR) = fMHz_low + 0.5 * d_fMHz
    END IF

    DUM_1 = ANG_hr_frq(i_HOUR,i_FRQ,1)

    IF ( DUM_1 .GT. 0.0 ) THEN

      DUM_2 = ANG_hr_frq(i_HOUR,i_FRQ,2)
      ANG_10(i_HOUR) = DUM_2 / DUM_1

    ELSE

" Determine average angle below current MUF10 frequency

      DUM_1 = 0.0
      DUM_2 = 0.0
      NONZERO_flag = .FALSE.
      DO j_FRQ = i_FRQ-1, 1, -1
        IF ( NONZERO_flag ) THEN
          CYCLE
        ELSE
          DUM_1 = ANG_hr_frq(i_HOUR,j_FRQ,1)
          IF ( DUM_1 .NE. 0.0 ) THEN
            NONZERO_flag = .TRUE.
            DUM_2 = ANG_hr_frq(i_HOUR,j_FRQ,2)
          END IF
        END IF
      END DO

" Determine average angle above current MUF10 frequency

      DUM_3 = 0.0
      DUM_4 = 0.0
```

```
      NONZERO_flag = .FALSE.
      DO j_FRQ = i_FRQ+1, N_FRQ
        IF ( NONZERO_flag ) THEN
          CYCLE
        ELSE
          DUM_3 = ANG_hr_frq(i_HOUR,j_FRQ,1)
          IF ( DUM_3 .NE. 0.0 ) THEN
            NONZERO_flag = .TRUE.
            DUM_4 = ANG_hr_frq(i_HOUR,j_FRQ,2)
          END IF
        END IF
      END DO

      DUM_5 = DUM_1 + DUM_3
      IF ( DUM_5 .NE. 0.0 ) THEN
        DUM_6 = DUM_2 + DUM_4
        ANG_10(i_HOUR) = DUM_6 / DUM_5
      END IF

    END IF

  END IF

  IF ( x2 .GT. 0.50 .AND. .NOT. P50_FLAG ) THEN

    P50_FLAG = .TRUE.

    x1 = fMHz_hr_bin(i_HOUR,i_FRQ)
    y1 = fMHz_low + ( FLOAT( i_FRQ - 1 ) + 0.5 ) * d_fMHz
      y2 = y1 + d_fMHz

    DUM_1 = x2 - x1

    IF ( DUM_1 .NE. 0.0 ) THEN
      m_MUF = ( y2 - y1 ) / DUM_1
      b_MUF = ( y1 * x2 - y2 * x1 ) / DUM_1
      DUM_2 = m_MUF * 0.50 + b_MUF
    ELSE
          DUM_2 = fMHz_low + ( FLOAT( i_FRQ - 1 ) + 0.5 ) * d_fMHz
    END IF

    IF ( DUM_2 .GE. fMHz_low ) THEN
      MUF_50(i_HOUR) = DUM_2
    ELSE
          MUF_50(i_HOUR) = fMHz_low + 0.5 * d_fMHz
    END IF
```

```
      DUM_1 = ANG_hr_frq(i_HOUR,i_FRQ,1)

      IF ( DUM_1 .GT. 0.0 ) THEN

        DUM_2 = ANG_hr_frq(i_HOUR,i_FRQ,2)
        ANG_50(i_HOUR) = DUM_2 / DUM_1

      ELSE
```

" Determine average angle below current MUF50 frequency

```
        DUM_1 = 0.0
        DUM_2 = 0.0
        NONZERO_flag = .FALSE.
        DO j_FRQ = i_FRQ-1, 1, -1
          IF ( NONZERO_flag ) THEN
            CYCLE
          ELSE
            DUM_1 = ANG_hr_frq(i_HOUR,j_FRQ,1)
            IF ( DUM_1 .NE. 0.0 ) THEN
              NONZERO_flag = .TRUE.
              DUM_2 = ANG_hr_frq(i_HOUR,j_FRQ,2)
            END IF
          END IF
        END DO
```

" Determine average angle above current MUF50 frequency

```
        DUM_3 = 0.0
        DUM_4 = 0.0
        NONZERO_flag = .FALSE.
        DO j_FRQ = i_FRQ+1, N_FRQ
          IF ( NONZERO_flag ) THEN
            CYCLE
          ELSE
            DUM_3 = ANG_hr_frq(i_HOUR,j_FRQ,1)
            IF ( DUM_3 .NE. 0.0 ) THEN
              NONZERO_flag = .TRUE.
              DUM_4 = ANG_hr_frq(i_HOUR,j_FRQ,2)
            END IF
          END IF
        END DO

        DUM_5 = DUM_1 + DUM_3
        IF ( DUM_5 .NE. 0.0 ) THEN
          DUM_6 = DUM_2 + DUM_4
          ANG_50(i_HOUR) = DUM_6 / DUM_5
```

```
      END IF

    END IF

  END IF

  IF ( x2 .GT. 0.90 .AND. .NOT. P90_FLAG ) THEN

    P90_FLAG = .TRUE.

    x1 = fMHz_hr_bin(i_HOUR,i_FRQ)
      y1 = fMHz_low + ( FLOAT( i_FRQ - 1 ) + 0.5 ) * d_fMHz
      y2 = y1 + d_fMHz

      DUM_1 = x2 - x1

    IF ( DUM_1 .NE. 0.0 ) THEN
      m_MUF = ( y2 - y1 ) / DUM_1
      b_MUF = ( y1 * x2 - y2 * x1 ) / DUM_1
      DUM_2 = m_MUF * 0.90 + b_MUF
    ELSE
        DUM_2 = fMHz_low + ( FLOAT( i_FRQ - 1 ) + 0.5 ) * d_fMHz
    END IF

    IF ( DUM_2 .GE. fMHz_low ) THEN
      MUF_90(i_HOUR) = DUM_2
    ELSE
        MUF_90(i_HOUR) = fMHz_low + 0.5 * d_fMHz
      END IF

      DUM_1 = ANG_hr_frq(i_HOUR,i_FRQ,1)

    IF ( DUM_1 .GT. 0.0 ) THEN

      DUM_2 = ANG_hr_frq(i_HOUR,i_FRQ,2)
      ANG_90(i_HOUR) = DUM_2 / DUM_1

    ELSE

* Determine average angle below current MUF90 frequency

      DUM_1 = 0.0
      DUM_2 = 0.0
      NONZERO_flag = .FALSE.
      DO j_FRQ = i_FRQ-1, 1, -1
        IF ( NONZERO_flag ) THEN
          CYCLE
```

```
            ELSE
              DUM_1 = ANG_hr_frq(i_HOUR,j_FRQ,1)
              IF ( DUM_1 .NE. 0.0 ) THEN
                NONZERO_flag = .TRUE.
                DUM_2 = ANG_hr_frq(i_HOUR,j_FRQ,2)
              END IF
            END IF
          END DO

* Determine average angle above current MUF90 frequency

          DUM_3 = 0.0
          DUM_4 = 0.0
          NONZERO_flag = .FALSE.
          DO j_FRQ = i_FRQ+1, N_FRQ
            IF ( NONZERO_flag ) THEN
              CYCLE
            ELSE
              DUM_3 = ANG_hr_frq(i_HOUR,j_FRQ,1)
              IF ( DUM_3 .NE. 0.0 ) THEN
                NONZERO_flag = .TRUE.
                DUM_4 = ANG_hr_frq(i_HOUR,j_FRQ,2)
              END IF
            END IF
          END DO

          DUM_5 = DUM_1 + DUM_3
          IF ( DUM_5 .NE. 0.0 ) THEN
            DUM_6 = DUM_2 + DUM_4
            ANG_90(i_HOUR) = DUM_6 / DUM_5
          END IF

        END IF

      END IF

      IF ( P10_FLAG .AND. P50_FLAG .AND. P90_FLAG ) CYCLE

    END DO

  END DO

* Output results by appending to disk file
* OUTPUT MUF values exceeded 90% of the time

  IF ( i_MONTH .EQ. 0 ) i_MONTH = MONTH_1
  WRITE(8,700) i_YEAR, i_MONTH, ( i, i = 0, 24 )
```

```
700  FORMAT(/2x, 12x, ' 10% NOT EXCEEDED (FOT)', 1x, i4, 1x, i2, -
           /2x, 6x, 25( 1x, i3, 1x ) )

     WRITE(8,702) 'FOT:  ', MUF_10(24), -
                  (MUF_10(i_HOUR),i_HOUR=1,24), -
                  'ANG:  ',ANG_10(24), -
                 (ANG_10(i_HOUR),i_HOUR=1,24)
702  FORMAT( 1x, a6, 25f5.1 /1x, a6, 25f5.1 )

*    OUTPUT MUF values exceeded 50% of the time

     WRITE(8,704) ( i, i = 0, 24 )
704  FORMAT(2x, 12x, ' 50% NOT EXCEEDED (MUF)', -
           /2x, 6x, 25( 1x, i3, 1x ) )

     WRITE(8,702) 'MUF:  ', MUF_50(24), -
                 (MUF_50(i_HOUR),i_HOUR=1,24), -
                     'ANG:  ', ANG_50(24), -
                 (ANG_50(i_HOUR),i_HOUR=1,24)

*    OUTPUT MUF values exceeded 10% of the time

     WRITE(8,708) ( i, i = 0, 24 )
708  FORMAT(2x, 12x, ' 90% NOT EXCEEDED (HPF)', -
           /2x, 6x, 25( 1x, i3, 1x ) )

     WRITE(8,702) 'HPF:  ', MUF_90(24), -
                 (MUF_90(i_HOUR),i_HOUR=1,24), -
                     'ANG:  ', ANG_90(24), -
                 (AN    _90(i_HOUR),i_HOUR=1,24)

     adg_bin = 0.0
     adg_hr_bin = 0.0
     adg_mode_bin = 0.0
     adg_mode_hr_bin = 0.0
     AVG_adg_mode_hr = 0.0
     AVG_adg_mode = 0.0
     ANG_hr_frq = 0.0
     dkm_bin = 0.0
     dkm_hr_bin = 0.0
     dkm_mode_bin = 0.0
     dkm_mode_hr_bin = 0.0
     AVG_dkm_mode_hr = 0.0
     AVG_dkm_mode = 0.0
     fMHz_bin = 0.0
     fMHz_hr_bin = 0.0
     fMHz_mode_bin = 0.0
```

```
      fMHz_mode_hr_bin = 0.0
      AVG_fMHz_mode_hr = 0.0
      SDV_fMHz_mode_hr = 0.0
      AVG_fMHz_mode = 0.0
      SDV_fMHz_mode = 0.0
      fMHz_hr_disp = 0.0
      fMHz_disp = 0.0
      N_dy = 0.0
      N_hr_dy = 0.0
      N_mode_dy = 0.0
      N_mode_hr_dy = 0.0

      MUF_10 = 0.0
      MUF_50 = 0.0
      MUF_90 = 0.0
      ANG_10 = 0.0
      ANG_50 = 0.0
      ANG_90 = 0.0

      RETURN

      END


*           Subroutine BEARING
*
*     Purpose:  Compute the latitude and longitude of two locations
*               in radians, the bearing angles from true north and
*               the Great Circle path distance.
*
      SUBROUTINE BEARING( STATION_name, COLAT_DG, LNG_DG, Ds_km, -
     &     As_rd, azimuth_dg, MONTHLY_flag )

      REAL*4 LAT_dg(2), LNG_dg(2), Ds_km, As_dg, azimuth_dg(2), -
     &     S_LAT(2), C_LAT(2), T_LAT(2), S_LNG(2), C_LNG(2), -
     &     T_LNG(2), B_dg(2), FLAG(2), S_dLNG(2), C_dLNG, DUM, -
     &     COLAT_dg(2), Re_km, Rearth_km

      CHARACTER*20 STATION_name(2)

      LOGICAL*1 MONTHLY_flag

      COMMON /MISC/ Re_km, Rearth_km

*     Define constants

      pi = 4.0 * ATAN( 1.0 )
```

```
      rddg = 180.0 / pi
      dgrd = pi / 180.0

*     Loop on each ionosonde station location

      DO i_SITE = 1, 2

        LAT_dg(i_SITE) = 90.0 - COLAT_dg(i_SITE)

        S_LAT(i_SITE) = SIN( dgrd * LAT_dg(i_SITE) )
        C_LAT(i_SITE) = COS( dgrd * LAT_dg(i_SITE) )
        T_LAT(i_SITE) = TAN( dgrd * LAT_dg(i_SITE) )

        S_LNG(i_SITE) = SIN( dgrd * LNG_dg(i_SITE) )
        C_LNG(i_SITE) = COS( dgrd * LNG_dg(i_SITE) )
        T_LNG(i_SITE) = TAN( dgrd * LNG_dg(i_SITE) )

      END DO

      C_dLNG   = COS( dgrd * ( LNG_dg(1) - LNG_dg(2) ) )
      S_dLNG(1) = SIN( dgrd * ( LNG_dg(2) - LNG_dg(1) ) )
      S_dLNG(2) = SIN( dgrd * ( LNG_dg(1) - LNG_dg(2) ) )

*     Path length in kilometers and Great Circle arc length

      DUM = C_LAT(1) * C_LAT(2) * C_dLNG + S_LAT(1) * S_LAT(2)
      Ds_km = Rearth_km * ACOS( DUM )
      As_rd = Ds_km / Rearth_km

*     Bearing angles from 1 to 2 and 2 to 1 measured clockwise from
*     true north

      DO i_SITE = 1, 2

        j_SITE = 2
        IF ( i_SITE .EQ. 2 ) j_SITE = 1

        DUM = C_LAT(i_SITE) * T_LAT(j_SITE) - S_LAT(i_SITE) * C_dLNG
        B_dg(i_SITE) = rddg * ATAN2( S_dLNG(i_SITE), DUM )

        FLAG(i_SITE) = SIGN( 1.0, S_dLNG(i_SITE) ) * SIGN( 1.0, B_dg(i_SITE) )

        IF ( FLAG(i_SITE) .GT. 0.0 ) THEN
          azimuth_dg(i_SITE) = B_dg(i_SITE)
        ELSE
          azimuth_dg(i_SITE) = B_dg(i_SITE) + 180.0
        END IF
```

```
      IF ( azimuth_dg(i_SITE) .LT. 0.0 ) THEN
        azimuth_dg(i_SITE) = azimuth_dg(i_SITE) + 360.0
      END IF

    END DO

    WRITE(6,700)
    WRITE(6,701)
    WRITE(6,703) ( i, STATION_name(i), COLAT_dg(i), LNG_dg(i), i = 1, 2 )
    WRITE(6,705) ( azimuth_dg(i), i = 1, 2 ), -
           2.0 * Ds_km, 2.0 * Ds_km / 1.609
    WRITE(6,700)

    IF ( MONTHLY_flag ) THEN

      WRITE(8,700)
      WRITE(8,701)
      WRITE(8,703) ( i, STATION_name(i), COLAT_dg(i), LNG_dg(i), -
             i = 1, 2 )
      WRITE(8,705) ( azimuth_dg(i), i = 1, 2 ), -
           2.0 * Ds_km, 2.0 * Ds_km / 1.609
      WRITE(8,700)

    ELSE

      WRITE(3,700)
      WRITE(3,701)
      WRITE(3,703) ( i, STATION_name(i), COLAT_dg(i), LNG_dg(i), -
             i = 1, 2 )
      WRITE(3,705) ( azimuth_dg(i), i = 1, 2 ), -
           2.0 * Ds_km, 2.0 * Ds_km / 1.609
      WRITE(3,700)

    END IF

700 FORMAT(//'------------------------------------------',-
       /'------------------------------------------')

701 FORMAT(/'       IONOSONDE STATION GEOGRAPHY ')

703 FORMAT(/' Site ', i2, 3x, a20 , -
          /11x,'    CoLatitude: ',f6.1,' DEG ', -
         /11x,' East Longitude: ',f6.1,' DEG ')

705 FORMAT(/'              Site 1-->Site 2  Site 2-->Site 1',-
       /' Bearing angles: ',f6.1,' DEG  ',5x,f6.1,' DEG',-
```

```
      /' Link Great Circle path length: ',f7.2,' KM', -
      /'                        or ',f7.2,' MI' )

RETURN

END
```

A.3.2 HFLINK2.FOR Source Program

```
$FREEFORM
$LARGE

"                    Subroutine HFLINK2
"
" Purpose: Process IONOSONDE data files from two stations following the
"          conventions established in UAG-23 and UAG-23A and determine
"          the optimum frequencies for all possible two-hop HF radio links
"          with reflection points above each ionosonde.

      SUBROUTINE HFLINK2( i_DAY, i_HOUR, LINK_flag, MONTHLY_flag )

" Declare variable types and dimension arrays

" Declare arrays for HF link calculations

      REAL*4 m_MUF, b_MUF, DUM_1, DUM_2, N_km_hr, N_km
      REAL*4 fMHz_low, fMHz_hgh, dkm_low, dkm_hgh
      REAL*4 fMHz_hr_disp, fMHz_disp
      REAL*4 adg_low, adg_hgh, d_fMHz, d_dkm
      REAL*4 a_limit, adg_bin, adg_hr_bin, adg_mode_bin
      REAL*4 adg_mode_hr_bin, AVG_adg_mode_hr, AVG_adg_mode
      REAL*4 d_limit, dkm_bin, dkm_hr_bin, dkm_mode_bin
      REAL*4 dkm_mode_hr_bin, AVG_dkm_mode_hr, AVG_dkm_mode
      REAL*4 f_limit, fMHz_bin, fMHz_hr_bin, fMHz_mode_bin
      REAL*4 fMHz_mode_hr_bin, AVG_fMHz_mode_hr, AVG_fMHz_mode
      REAL*4 SDV_fMHz_mode_hr, SDV_fMHz_mode, N_noLOS
      REAL*4 N_dy, N_hr_dy, N_mode_dy, N_mode_hr_dy, N_dy_hr, N_hr, N_GOOD
      REAL*4 LAT_dg(2), LNG_dg(2), Ds_km, azimuth_dg(2), Rearth_km
      REAL*4 hv_km(4,4,2), fo_MHz(4,4,2), Re_km, alfa_rd(2), MUF_MHz(2)
      REAL*4 f_MUF_MHz(4,4), d_MUF_km(4,4,2), a_MUF_dg(4,4)
      REAL h_LINK_km, f_LINK_MHz, a_LINK_dg, D_km, ANG_hr_frq
      REAL*4 MUF_fctr, hF2_km, hF1_km, hE_km, hEs_km
      REAL*4 Kf, k_VALUE, r_km(2), phi_rd(2), TO_rd(2)
      REAL*4 foF2, fxF2, fzF2, M3000F2, hF2, hpf2, foF1, fxF1, -
             M3000F1, hF1, hF, foE, foE2, hE, hE2, foEs, -
             fxEs, fbEs, fEs, hEs, foF1d5, fmin, M3000F1d5, hf1d5, -
             foI, fxI, fmI, I2000, II, Ixxx

      INTEGER*4 i_DAY, i_HOUR, i_DST, i_FRQ, i_MODE, i_SITE
      INTEGER*4 N_DST, N_FRQ, N_ANG
      INTEGER*4 N_bins, month_OLD
      INTEGER*4 i_YEAR, i_MONTH
      INTEGER*4 ONE, N_DISP
```

```
      CHARACTER*5 L_heading(3), MODE_name(1:4,1:4)
      CHARACTER*25 LNK_file
      CHARACTER*25 DIST_heading

      LOGICAL*1 P10_FLAG, P50_FLAG, P90_FLAG, MONTHLY_flag
      LOGICAL*1 FAIL_flag, MODE_flag(4,4,2), NONZERO_flag
      LOGICAL*1 INTV, ITRN, D_flag, LINK_flag(4,4), GOOD_LINK
      LOGICAL*1 TEST_FLAG

C     ***************************************************************************
C     Define common blocks

      COMMON/IONSNP/ foF2(2,31,24), M3000F2(2,31,24), hF2(2,31,24), -
                     foF1(2,31,24), M3000F1(2,31,24), hF1(2,31,24), -
                     hF(2,31,24), foE(2,31,24), -
                     hE(2,31,24), foEs(2,31,24), fbEs(2,31,24), -
                         fEs(2,31,24), hEs(2,31,24)

C                        fmin(2,31,24), fxI(2,31,24)

      COMMON /MISC/ Re_km, Rearth_km

      COMMON /LIMIT/ fMHz_low, fMHz_hgh, N_FRQ, dkm_low, dkm_hgh, N_DST, -
                     adg_low, adg_hgh, N_ANG, LNK_file, d_fMHz, d_dkm, -
                     d_adg, As_rd, Ds_km

C     COMMON /ANGLE/ adg_bin(0:61), -
C                    adg_hr_bin(1:24,0:61), adg_mode_bin(1:4,1:4,0:61), -
C                    adg_mode_hr_bin(1:4,1:4,1:24,0:61), -
C                    AVG_adg_mode_hr(1:4,1:4,1:24), AVG_adg_mode(1:4,1:4), -
C                    ANG_hr_frq(1:24,0:61,1:2)

      COMMON /ANGLE/ ANG_hr_frq(1:24,0:61,1:2)

C     COMMON /RANGE/ dkm_bin(0:61), -
C                    dkm_hr_bin(1:24,0:61), dkm_mode_bin(1:4,1:4,0:61), -
C                    dkm_mode_hr_bin(1:4,1:4,1:24,0:61), -
C                    AVG_dkm_mode_hr(1:4,1:4,1:24), AVG_dkm_mode(1:4,1:4)

      COMMON /FRQBN/ f_limit(1:12,1:2), fMHz_bin(0:61), -
                     fMHz_hr_bin(1:24,0:61), fMHz_mode_bin(1:4,1:4,0:61), -
                     fMHz_mode_hr_bin(1:4,1:4,1:24,0:61), -
                     AVG_fMHz_mode_hr(1:4,1:4,1:24), -
                     SDV_fMHz_mode_hr(1:4,1:4,1:24), -
                         AVG_fMHz_mode(1:4,1:4), SDV_fMHz_mode(1:4,1:4), -
                         fMHz_hr_disp(1:24,1:12), fMHz_disp(1:12)
```

```
      COMMON /COUNT/ N_dy, N_hr_dy(1:24), N_mode_dy(1:4,1:4), N_GOOD, -
                N_mode_hr_dy(1:4,1:4,1:24), N_dy_hr(1:24), N_hr, -
                N_noLOS

*  ************************************************************************

      DATA N_DISP/12/

*  ************************************************************************
*  ************************************************************************

*  Compute constants

      S_3000 = SIN( 3000.0 / ( 2.0 * Rearth_km ) )
      C_3000 = COS( 3000.0 / ( 2.0 * Rearth_km ) )
      pi = 4.0 * ATAN( 1.0 )
      rddg = 180.0 / pi
      dgrd = pi / 180.0

      hv_km = -1.0
      fo_MHz = -1.0
      f_MUF_MHz = -1.0
      a_LINK_dg = 0.0
      a_MUF_dg = 0.0
      d_LINK_km = 0.0
      d_MUF_km = 0.0

      MODE_flag = .FALSE.
      LINK_flag = .FALSE.

*  DETERMINE REFLECTION HEIGHTS FOR EACH AVAILABLE MODE

      DO i_HOP = 1, 2

*        WRITE(6,*) i_YEAR, i_MONTH, i_DAY, i_HOUR, i_HOP
*        WRITE(6,*) 'foF2: ', foF2(i_HOP,i_DAY,i_HOUR)

        IF ( foF2(i_HOP,i_DAY,i_HOUR) .GT. 0.0 ) THEN

          IF ( M3000F2(i_HOP,i_DAY,i_HOUR) .GT. 0.0 ) THEN
            MUF_fctr = M3000F2(i_HOP,i_DAY,i_HOUR)
            M3000F2(i_HOP,i_DAY,i_HOUR) = -1.0
            DUM_1 = MUF_fctr / k_VALUE( 3000.0 )
            DUM_1 = SQRT( DUM_1**2 - 1 )
            hF2_km = Rearth_km * ( S_3000 / DUM_1 + C_3000 - 1.0 )
          ELSE IF ( hF2(i_HOP,i_DAY,i_HOUR) .GT. 0.0 ) THEN
              hF2_km = hF2(i_HOP,i_DAY,i_HOUR)
```

```
        hF2(i_HOP,i_DAY,i_HOUR) = -1.0
      ELSE
       hF2_km = 325.0
      END IF

      DO j_MODE = 1, 4

       IF ( i_HOP .EQ. 1 ) THEN
            MODE_flag(1,j_MODE,1) = .TRUE.
            fo_MHz(1,j_MODE,1) = foF2(1,i_DAY,i_HOUR)
         hv_km(1,j_MODE,1) = hF2_km
       END IF

       IF ( i_HOP .EQ. 2 ) THEN
            MODE_flag(j_MODE,1,2) = .TRUE.
            fo_MHz(j_MODE,1,2) = foF2(2,i_DAY,i_HOUR)
            hv_km(j_MODE,1,2) = hF2_km
       END IF

      END DO

      foF2(i_HOP,i_DAY,i_HOUR) = -1.0

     ELSE

      DO j_MODE = 1, 4

       IF ( i_HOP .EQ. 1 ) THEN
            MODE_flag(1,j_MODE,1) = .FALSE.
         fo_MHz(1,j_MODE,1) = -1.0
         hv_km(1,j_MODE,1) = -1.0
       END IF

       IF ( i_HOP .EQ. 2 ) THEN
            MODE_flag(j_MODE,1,2) = .FALSE.
         fo_MHz(j_MODE,1,2) = -1.0
         hv_km(j_MODE,1,2) = -1.0
       END IF

      END DO

     END IF

      WRITE(6,*) '  foF1: ', foF1(i_HOP,i_DAY,i_HOUR)

     IF ( foF1(i_HOP,i_DAY,i_HOUR) .GT. 0.0 ) THEN
```

```
IF ( M3000F1(i_HOP,i_DAY,i_HOUR) .GT. 0.0 ) THEN
  MUF_fctr = M3000F1(i_HOP,i_DAY,i_HOUR)
  M3000F1(i_HOP,i_DAY,i_HOUR) = -1.0
  DUM_1 = MUF_fctr / k_VALUE( 3000.0 )
  DUM_1 = SQRT( DUM_1**2 - 1 )
  DUM_2 = S_3000 / DUM_1 + C_3000 - 1.0
  hF1_km = Rearth_km * ( DUM_2 )
ELSE IF ( hF1(i_HOP,i_DAY,i_HOUR) .GT. 0.0 ) THEN
     hF1_km = hF1(i_HOP,i_DAY,i_HOUR)
     hF1(i_HOP,i_DAY,i_HOUR) = -1.0
ELSE IF ( hF(i_HOP,i_DAY,i_HOUR) .GT. 0.0 ) THEN
     hF1_km = hF(i_HOP,i_DAY,i_HOUR)
     hF(i_HOP,i_DAY,i_HOUR) = -1.0
ELSE
  hF1_km = 200.0
END IF

DO j_MODE = 1, 4

 IF ( i_HOP .EQ. 1 ) THEN
       MODE_flag(2,j_MODE,1) = .TRUE.
       fo_MHz(2,j_MODE,1) = foF1(1,i_DAY,i_HOUR)
   hv_km(2,j_MODE,1) = hF1_km
 END IF

 IF ( i_HOP .EQ. 2 ) THEN
       MODE_flag(j_MODE,2,2) = .TRUE.
       fo_MHz(j_MODE,2,2) = foF1(2,i_DAY,i_HOUR)
       hv_km(j_MODE,2,2) = hF1_km
 END IF

END DO

foF1(i_HOP,i_DAY,i_HOUR) = -1.0

ELSE

DO j_MODE = 1, 4

 IF ( i_HOP .EQ. 1 ) THEN
       MODE_flag(2,j_MODE,1) = .FALSE.
   fo_MHz(2,j_MODE,1) = -1.0
   hv_km(2,j_MODE,1) = -1.0
 END IF

 IF ( i_HOP .EQ. 2 ) THEN
       MODE_flag(j_MODE,2,2) = .FALSE.
```

```
          fo_MHz(j_MODE,2,2) = -1.0
          hv_km(j_MODE,2,2) = -1.0
        END IF

      END DO

    END IF

*     WRITE(6,*) '      foE: ', foE(i_HOP,i_DAY,i_HOUR)

    IF ( foE(i_HOP,i_DAY,i_HOUR) .GT. 0.0 ) THEN

      IF ( hE(i_HOP,i_DAY,i_HOUR) .GT. 0.0 ) THEN
        hE_km = hE(i_HOP,i_DAY,i_HOUR)
        hE(i_HOP,i_DAY,i_HOUR) = -1.0
      ELSE
        hE_km = 110.0
      END IF

      DO j_MODE = 1, 4

        IF ( i_HOP .EQ. 1 ) THEN
              MODE_flag(3,j_MODE,1) = .TRUE.
          fo_MHz(3,j_MODE,1) = foE(1,i_DAY,i_HOUR)
          hv_km(3,j_MODE,1) = hE_km
        END IF

        IF ( i_HOP .EQ. 2 ) THEN
              MODE_flag(j_MODE,3,2) = .TRUE.
          fo_MHz(j_MODE,3,2) = foE(2,i_DAY,i_HOUR)
              hv_km(j_MODE,3,2) = hE_km
        END IF

      END DO

      foE(i_HOP,i_DAY,i_HOUR) = -1.0

    ELSE

      DO j_MODE = 1, 4

        IF ( i_HOP .EQ. 1 ) THEN
              MODE_flag(3,j_MODE,1) = .FALSE.
          fo_MHz(3,j_MODE,1) = -1.0
          hv_km(3,j_MODE,1) = -1.0
        END IF
```

```
      IF ( i_HOP .EQ. 2 ) THEN
            MODE_flag(j_MODE,3,2) = .FALSE.
        fo_MHz(j_MODE,3,2) = -1.0
        hv_km(j_MODE,3,2) = -1.0
      END IF

    END DO

  END IF

*   WRITE(6,*) '          foEs: ', foEs(i_HOP,i_DAY,i_HO

  IF ( foEs(i_HOP,i_DAY,i_HOUR) .GT. 0.0 ) THEN

    IF ( hEs(i_HOP,i_DAY,i_HOUR) .GT. 0.0 ) THEN
      hEs_km = hEs(i_HOP,i_DAY,i_HOUR)
      hEs(i_HOP,i_DAY,i_HOUR) = -1.0
    ELSE
      hEs_km = 120.0
    END IF

    DO j_MODE = 1, 4

      IF ( i_HOP .EQ. 1 ) THEN
            MODE_flag(4,j_MODE,1) = .TRUE.
            fo_MHz(4,j_MODE,1) = foEs(1,i_DAY,i_HOUR)
        hv_km(4,j_MODE,1) = hEs_km
      END IF

      IF ( i_HOP .EQ. 2 ) THEN
            MODE_flag(j_MODE,4,2) = .TRUE.
            fo_MHz(j_MODE,4,2) = foEs(2,i_DAY,i_HOUR)
            hv_km(j_MODE,4,2) = hEs_km
      END IF

    END DO

    foEs(i_HOP,i_DAY,i_HOUR) = -1.0

  ELSE

    DO j_MODE = 1, 4

      IF ( i_HOP .EQ. 1 ) THEN
            MODE_flag(4,j_MODE,1) = .FALSE.
        fo_MHz(4,j_MODE,1) = -1.0
        hv_km(4,j_MODE,1) = -1.0
```

```
            END IF

            IF ( i_HOP .EQ. 2 ) THEN
                  MODE_flag(j_MODE,4,2) = .FALSE.
              fo_MHz(j_MODE,4,2) = -1.0
              hv_km(j_MODE,4,2) = -1.0
            END IF

          END DO

        END IF

"       WRITE(6,*)
"       WRITE(6,777) 'fo_MHz(1,*,i_HOP): ',(fo_MHz(1,i,i_HOP),i=1,4)
"       WRITE(6,777) 'fo_MHz(2,*,i_HOP): ',(fo_MHz(2,i,i_HOP),i=1,4)
"       WRITE(6,777) 'fo_MHz(3,*,i_HOP): ',(fo_MHz(3,i,i_HOP),i=1,4)
"       WRITE(6,777) 'fo_MHz(4,*,i_HOP): ',(fo_MHz(4,i,i_HOP),i=1,4)
"       WRITE(6,*)
"       WRITE(6,777) 'hv_km(1,*,i_HOP): ',(hv_km(1,i,i_HOP),i=1,4)
"       WRITE(6,777) 'hv_km(2,*,i_HOP): ',(hv_km(2,i,i_HOP),i=1,4)
"       WRITE(6,777) 'hv_km(3,*,i_HOP): ',(hv_km(3,i,i_HOP),i=1,4)
"       WRITE(6,777) 'hv_km(4,*,i_HOP): ',(hv_km(4,i,i_HOP),i=1,4)
"       WRITE(6,777)
"       WRITE(6,*) 'MODE_flag(1,*,i_HOP): ',(MODE_flag(1,i,i_HOP),i=1,4)
"       WRITE(6,*) 'MODE_flag(2,*,i_HOP): ',(MODE_flag(2,i,i_HOP),i=1,4)
"       WRITE(6,*) 'MODE_flag(3,*,i_HOP): ',(MODE_flag(3,i,i_HOP),i=1,4)
"       WRITE(6,*) 'MODE_flag(4,*,i_HOP): ',(MODE_flag(4,i,i_HOP),i=1,4)
"       WRITE(6,*)

      END DO

777   FORMAT(1x,a20,4(f6.1,1x))

" Exit if no two-hop propagation is possible.

      K_COUNT = 0
      DO i_MODE = 1, 4
          DO j_MODE = 1, 4
            LINK_flag(i_MODE,j_MODE) = MODE_flag(i_MODE,j_MODE,1) -
                          .AND. MODE_flag(i_MODE,j_MODE,2)
          IF ( LINK_flag(i_MODE,j_MODE) ) K_COUNT = K_COUNT + 1
          END DO
      END DO

"     IF ( K_COUNT .GT. 0.0 ) N_GOOD = N_GOOD + 1.0

"         WRITE(6,*) 'LINK_flag(1,*): ',(LINK_flag(1,i),i=1,4)
```

```
"       WRITE(6,*) 'LINK_flag(2,*): ',(LINK_flag(2,i),i=1,4)
"       WRITE(6,*) 'LINK_flag(3,*): ',(LINK_flag(3,i),i=1,4)
"       WRITE(6,*) 'LINK_flag(4,*): ',(LINK_flag(4,i),i=1,4)
"       WRITE(6,*)

" Determine the propagation range achieved for each reflection height.

    f_LINK_MHz = 10000.0
    MUF_MHz = -1.0
    GOOD_LINK = .FALSE.

    N_LOS = 0

    DO i_MODE = 1, 4

      DO j_MODE = 1, 4

" Drop out of loop if current two-hop mode does not have adequate data.

        IF ( .NOT. LINK_flag(i_MODE,j_MODE) ) CYCLE

        N_LOS = N_LOS + 1

" Determine if overlapping radio LOS is achieved given the two virtual
" reflection heights.

        r_km(1) = Rearth_km + hv_km(i_MODE,j_MODE,1)
        a1_LOS_rd = ACOS( Rearth_km / r_km(1) )

        r_km(2) = Rearth_km + hv_km(i_MODE,j_MODE,2)
        a2_LOS_rd = ACOS( Rearth_km / r_km(2) )

         IF ( a1_LOS_rd + a2_LOS_rd .LT. As_rd ) THEN
           LINK_flag(i_MODE,j_MODE) = .FALSE.
         N_LOS = N_LOS - 1
        END IF

         IF( .NOT. LINK_flag(i_MODE,j_MODE) ) CYCLE

" Determine the ground reflection point where the angle of incidence is
" equal to the angle of reflection.  This calculation requires root finding
" and employs a modified Newton's method.

         a_LOW_rd = 0.0
         IF ( a2_LOS_rd .LT. As_rd ) a_LOW_rd = As_rd - a2_LOS_rd
        a_HGH_rd = a1_LOS_rd
        IF ( a1_LOS_rd .GT. As_rd ) a_HGH_rd = As_rd
```

```
theta_rd = ROOT( a_LOW_rd, a_HGH_rd, r_km, As_rd, INTV, ITRN )
IF ( .NOT. INTV .OR. .NOT. ITRN ) THEN
    LINK_flag(i_MODE,j_MODE) = .FALSE.
  END IF
IF ( .NOT. LINK_flag(i_MODE,j_MODE) ) CYCLE

alfa_rd(1) = 2.0 * theta_rd
alfa_rd(2) = 2.0 * As_rd - alfa_rd(1)

  GOOD_LINK = .TRUE.

  DO i_HOP = 1, 2

  DUM_1 = SIN( alfa_rd(i_HOP) / 2.0 )
  DUM_2 = 1 + hv_km(i_MODE,j_MODE,i_HOP) / Rearth_km -
        - COS( alfa_rd(i_HOP) / 2.0 )

  IF ( DUM_2 .LT. 1.0 ) THEN

    IF ( 1.0e38 * DUM_2 .GT. DUM_1 ) THEN
      phi_rd(i_HOP) = ATAN2( DUM_1, DUM_2 )
    ELSE
      phi_rd(i_HOP) = pi / 2.0
    END IF

  ELSE

    phi_rd(i_HOP) = ATAN2( DUM_1, DUM_2 )

  END IF

  TO_rd(i_HOP) = ( pi - alfa_rd(i_HOP) ) / 2.0 - phi_rd(i_HOP)

" Compute the "secant corrected" factor k.

  d_GND_km = Rearth_km * alfa_rd(i_HOP)
    IF ( i_HOP .EQ. 1 ) d_MUF_km(i_MODE,j_MODE,1) = d_GND_km
  Kf = k_VALUE( d_GND_km )

" Compute the maximum usable frequency for the the current hop.

    MUF_MHz(i_HOP) = Kf * fo_MHz(i_MODE,j_MODE,i_HOP) -
           / COS( phi_rd(i_HOP) )

END DO
```

```
        IF ( MUF_MHz(1) .LT. 0.0 .OR. MUF_MHz(2) .LT. 0.0 ) CYCLE

"  Two-hop MUF is the minimum of the MUFs of the two hops and the
"  two-hop TO angle is given by the corresponding value of alfa.

"         WRITE(6,*) 'As_dg: ', rddg*As_rd,' theta_dg: ', rddg*theta_rd
"         WRITE(6,*) 'alfa_dg(1): ', rddg*alfa_rd(1), -
"                ' alfa_dg(2): ', rddg*alfa_rd(2)
"         WRITE(6,*) 'phi_dg(1): ', rddg*phi_rd(1), -
"                ' phi_dg(2): ', rddg*phi_rd(2)
"         WRITE(6,*) 'TO_dg(1): ', rddg*TO_rd(1), -
"                ' TO_dg(2): ', rddg*TO_rd(2)
"         WRITE(6,*) '-----------------------------------------'

        IF ( MUF_MHz(1) .LT. MUF_MHz(2) ) THEN
          f_MUF_MHz(i_MODE,j_MODE) = MUF_MHz(1)
          a_MUF_dg(i_MODE,j_MODE) = rddg * TO_rd(1)
        ELSE
          f_MUF_MHz(i_MODE,j_MODE) = MUF_MHz(2)
          a_MUF_dg(i_MODE,j_MODE) = rddg * TO_rd(2)
        END IF

"  The overall MUF is the minimum of all the usable MUF values.

          IF ( f_MUF_MHz(i_MODE,j_MODE) .LT. f_LINK_MHz ) THEN
          f_LINK_MHz = f_MUF_MHz(i_MODE,j_MODE)
          a_LINK_dg = a_MUF_dg(i_MODE,j_MODE)
          d_LINK_km = d_MUF_km(i_MODE,j_MODE,1)
        END IF

        END DO

      END DO

      IF ( f_LINK_MHz .GT. 1000.0 ) GOOD_LINK = .FALSE.

"     WRITE(6,*)
"     WRITE(6,*) 'GOOD_LINK:  ', GOOD_LINK
"     WRITE(6,777) 'f_MUF_MHz(1,j): ', (f_MUF_MHz(1,j),j=1,4)
"     WRITE(6,777) 'f_MUF_MHz(2,j): ', (f_MUF_MHz(2,j),j=1,4)
"     WRITE(6,777) 'f_MUF_MHz(3,j): ', (f_MUF_MHz(3,j),j=1,4)
"     WRITE(6,777) 'f_MUF_MHz(4,j): ', (f_MUF_MHz(4,j),j=1,4)
"     WRITE(6,*) 'f_LINK_MHz:  ', f_LINK_MHz

"  INCREMENT THE APPROPRIATE NUMBER DENSITY BINS.

"  Exit if no link was possible.
```

```
      IF ( .NOT  GOOD_LINK ) THEN
        IF ( N_LOS .LE. 0 ) N_noLOS = N_noLOS + 1.0
          f_LINK_MHz = 0.0
          RETURN
      END IF

*     Total number of hours with usable data, the frequencies, angles, and
*     distances.

*       N_GOOD = N_GOOD + 1.0

        N_dy = N_dy + 1.0

        CALL BINS_0( f_LINK_MHz, fMHz_low, fMHz_hgh, fMHz_bin, N_FRQ )
*       CALL BINS_0( a_LINK_dg, adg_low, adg_hgh, adg_bin, N_ANG )
*       CALL BINS_0( d_LINK_km, dkm_low, dkm_hgh, dkm_bin, N_DST )

*     Increment the number of days for which data is available in the current
*     hour.

        N_hr_dy(i_HOUR) = N_hr_dy(i_HOUR) + 1.0

        CALL BINS_1( i_HOUR, f_LINK_MHz, fMHz_low, fMHz_hgh, 24, -
                     fMHz_hr_bin, N_FRQ )

*       CALL BINS_1( i_HOUR, a_LINK_dg, adg_low, adg_hgh, 24, -
*                    adg_hr_bin, N_ANG )
*       CALL BINS_1( i_HOUR, d_LINK_km, dkm_low, dkm_hgh, 24, -
*                    dkm_hr_bin, N_DST )

*     Increment the number of days for which data is available in the current
*     hour for each available two-hop propagation mode.

        DO i_MODE = 1, 4

          DO j_MODE = 1, 4

            IF ( .NOT. LINK_flag(i_MODE,j_MODE) ) CYCLE

            N_mode_dy(i_MODE,j_MODE) = N_mode_dy(i_MODE,j_MODE) + 1.0
            N_mode_hr_dy(i_MODE,j_MODE,i_HOUR) = -
                                 N_mode_hr_dy(i_MODE,j_MODE,i_HOUR) + 1.0

            AVG_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR) = -
              AVG_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR) + f_LINK_MHz
```

```
      DUM = f_LINK_MHz * f_LINK_MHz
      SDV_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR) = -
          SDV_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR) + DUM

*     IF ( i_MODE .EQ. 1 .AND. j_MODE .EQ. 1 .AND. i_HOUR .EQ. 1 ) THEN
*       WRITE(6,*) i_MODE, j_MODE, i_HOUR, f_LINK_MHz
*       WRITE(6,*) 'AVG: ', AVG_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR)
*       WRITE(6,*) 'SDV: ', SDV_fMHz_mode_hr(i_MODE,j_MODE,i_HOUR)
*     END IF

*     AVG_adg_mode_hr(i_MODE,j_MODE,i_HOUR) = -
*         AVG_adg_mode_hr(i_MODE,j_MODE,i_HOUR) + a_LINK_dg
*     AVG_dkm_mode_hr(i_MODE,j_MODE,i_HOUR) = -
*         AVG_dkm_mode_hr(i_MODE,j_MODE,i_HOUR) + d_LINK_km

      AVG_fMHz_mode(i_MODE,j_MODE) = -
              AVG_fMHz_mode(i_MODE,j_MODE) + f_LINK_MHz
      SDV_fMHz_mode(i_MODE,j_MODE) = -
              SDV_fMHz_mode(i_MODE,j_MODE) + DUM
*     AVG_adg_mode(i_MODE,j_MODE) = -
*             AVG_adg_mode(i_MODE,j_MODE) + a_LINK_dg
*     AVG_dkm_mode(i_MODE,j_MODE) = -
*         AVG_dkm_mode(i_MODE,j_MODE) + d_LINK_km

       CALL BINS_2( i_MODE, j_MODE, f_MUF_MHz(i_MODE,j_MODE), -
                    fMHz_low, fMHz_hgh, 4, 4, fMHz_mode_bin, -
              N_FRQ )
*     CALL BINS_2( i_MODE, j_MODE, a_MUF_dg(i_MODE,j_MODE), -
*              adg_low, adg_hgh, 4, 4, adg_mode_bin, N_ANG )
*     CALL BINS_2( i_MODE, j_MODE, d_MUF_km(i_MODE,j_MODE,1), -
*              dkm_low, dkm_hgh, 4, 4, dkm_mode_bin, N_DST )

       CALL BINS_3( i_MODE, j_MODE, i_HOUR, f_MUF_MHz(i_MODE,j_MODE), -
                    fMHz_low, fMHz_hgh, 4, 4, 24, fMHz_mode_hr_bin, -
              N_FRQ )
*     CALL BINS_3( i_MODE, j_MODE, i_HOUR, a_MUF_dg(i_MODE,j_MODE), -
*              adg_low, adg_hgh, 4, 4, 24, adg_mode_hr_bin, -
*              N_ANG )
*     CALL BINS_3( i_MODE, j_MODE, i_HOUR, d_MUF_km(i_MODE,j_MODE,1), -
*              dkm_low, dkm_hgh, 4, 4, 24, dkm_mode_hr_bin, -
*              N_DST )

    END DO

  END DO

  DO i_DISP = 1, N_DISP
```

```
      IF ( f_LINK_MHz .GT. f_limit(i_DISP,1) -
         .AND. f_LINK_MHz .LE. f_limit(i_DISP,2) ) THEN

        fMHz_hr_disp(i_HOUR,i_DISP) = fMHz_hr_disp(i_HOUR,i_DISP) + 1.0
        fMHz_disp(i_DISP) = fMHz_disp(i_DISP) + 1.0

      END IF

    END DO

*   Update frequency-conditional take-off angle mean bins.

    i_FRQ = INT( ( f_LINK_MHz - fMHz_low ) / d_fMHz ) + 1

    IF ( f_LINK_MHz .LE. fMHz_low ) THEN

      ANG_hr_frq(i_HOUR,0,1) = ANG_hr_frq(i_HOUR,0,1) + 1.0
      ANG_hr_frq(i_HOUR,0,2) = ANG_hr_frq(i_HOUR,0,2) + a_LINK_dg

    ELSE IF ( f_LINK_MHz .GT. fMHz_hgh ) THEN

        ANG_hr_frq(i_HOUR,N_FRQ+1,1) = ANG_hr_frq(i_HOUR,N_FRQ+1,1) -
                                     + 1.0
        ANG_hr_frq(i_HOUR,N_FRQ+1,2) = ANG_hr_frq(i_HOUR,N_FRQ+1,2) -
                                     + a_LINK_dg

    ELSE

      ANG_hr_frq(i_HOUR,i_FRQ,1) = ANG_hr_frq(i_HOUR,i_FRQ,1) + 1.0
      ANG_hr_frq(i_HOUR,i_FRQ,2) = ANG_hr_frq(i_HOUR,i_FRQ,2) + a_LINK_dg

    END IF

    RETURN

    END



*                   SUBROUTINE k_VALUE
*
*   Purpose:  To compute the 'secant-corrected' effect of a curved ionosphere


    REAL*4 FUNCTION k_VALUE( D )
```

```
      REAL*4 D, D_smp(17), k_smp(17), m_k, b_k

      DATA D_smp/0.0,231.9,318.8,376.8,492.8,608.7,782.6,985.5,1246.4,-
     &     2800.0,3014.5,3205.8,3373.9,3536.2,3687.0,3884.1,-
     &     4000.0/
      DATA k_smp/1.0,1.0,1.0007,1.0014,1.0035,1.0058,1.0111,1.0188,-
     &     1.0294,1.1047,1.1153,1.1259,1.1365,1.1471,1.1612,-
     &     1.1772,1.1871/

      IF ( D .LT. D_smp(1) ) THEN
        k_VALUE = k_smp(1)
        RETURN
      END IF

      IF ( D .GE. D_smp(17) ) THEN
        k_VALUE = k_smp(17)
        RETURN
      END IF

      DO k = 2, 17

        IF ( D_smp(k-1) .LE. D .AND. D .LT. D_smp(k) ) THEN

          DUM_1 = D_smp(k) - D_smp(k-1)
          m_k = ( k_smp(k) - k_smp(k-1) ) / DUM_1
          b_k = ( k_smp(k-1) * D_smp(k) - k_smp(k) * D_smp(k-1) ) / DUM_1
          k_VALUE = m_k * D + b_k
          RETURN

        END IF

      END DO

      RETURN

      END




"                    SUBROUTINE binFIL0
"
" Purpose:  Increment density function bin corresponding to input value

      SUBROUTINE BINS_0( Vr, Vr_low, Vr_hgh, bin_0, N_bins )
```

```
      REAL*4 bin_0(0:61)
      REAL*4 Vr, Vr_low, Vr_hgh, d_Vr
      INTEGER*4 N_bins

      LOGICAL*1 B_flag

      IF ( Vr .LE. Vr_low ) THEN
          bin_0(0) = bin_0(0) + 1.0
      ELSE IF ( Vr .GT. Vr_hgh ) THEN
            bin_0(N_bins+1) = bin_0(N_bins+1) + 1.0
      ELSE
        i_bin = 1
        B_flag = .FALSE.
          d_Vr = ( Vr_hgh - Vr_low ) / FLOAT( N_bins )
        DO WHILE ( i_bin .LE. N_bins .AND. .NOT. B_flag )
            Vr_min = Vr_low + ( i_bin - 1 ) * d_Vr
          Vr_max = Vr_min + d_Vr
          IF ( Vr .GT. Vr_min .AND. Vr .LE. Vr_max ) THEN
              bin_0(i_bin) = bin_0(i_bin) + 1.0
              B_flag = .TRUE.
          END IF
          i_bin = i_bin + 1
        END DO
      END IF

      RETURN

      END


"                    SUBROUTINE binFIL1
"
" Purpose:  Increment density function bin corresponding to input value

      SUBROUTINE BINS_1( i_1, Vr, Vr_low, Vr_hgh, N_1, bin_1, N_bins )

      REAL*4 bin_1(1:24,0:61)
      REAL*4 Vr, Vr_low, Vr_hgh, d_Vr
      INTEGER*4 N_bins, i_1, N_1

      LOGICAL*1 B_flag

      IF ( Vr .LE. Vr_low ) THEN
          bin_1(i_1,0) = bin_1(i_1,0) + 1.0
      ELSE IF ( Vr .GT. Vr_hgh ) THEN
            bin_1(i_1,N_bins+1) = bin_1(i_1,N_bins+1) + 1.0
```

```
      ELSE
        i_bin = 1
        B_flag = .FALSE.
          d_Vr = ( Vr_hgh - Vr_low ) / FLOAT( N_bins )
        DO WHILE ( i_bin .LE. N_bins .AND. .NOT. B_flag )
            Vr_min = Vr_low + ( i_bin - 1 ) * d_Vr
          Vr_max = Vr_min + d_Vr
          IF ( Vr .GT. Vr_min .AND. Vr .LE. Vr_max ) THEN
              bin_1(i_1,i_bin) = bin_1(i_1,i_bin) + 1.0
              B_flag = .TRUE.
          END IF
          i_bin = i_bin + 1
        END DO
      END IF

      RETURN

      END



*                     SUBROUTINE binFIL2
*
* Purpose:  Increment density function bin corresponding to input value

      SUBROUTINE BINS_2( i_1, i_2, Vr, Vr_low, Vr_hgh, N_1, N_2, -
                         bin_2, N_bins )

      REAL*4 bin_2(1:4,1:4,0:61)
      REAL*4 Vr, Vr_low, Vr_hgh, d_Vr
      INTEGER*4 N_bins, i_1, i_2, N_1, N_2

      LOGICAL*1 B_flag

      IF ( Vr .LE. Vr_low ) THEN
          bin_2(i_1,i_2,0) = bin_2(i_1,i_2,0) + 1.0
      ELSE IF ( Vr .GT. Vr_hgh ) THEN
            bin_2(i_1,i_2,N_bins+1) = bin_2(i_1,i_2,N_bins+1) + 1.0
      ELSE
        i_bin = 1
        B_flag = .FALSE.
          d_Vr = ( Vr_hgh - Vr_low ) / FLOAT( N_bins )
        DO WHILE ( i_bin .LE. N_bins .AND. .NOT. B_flag )
            Vr_min = Vr_low + ( i_bin - 1 ) * d_Vr
          Vr_max = Vr_min + d_Vr
          IF ( Vr .GT. Vr_min .AND. Vr .LE. Vr_max ) THEN
              bin_2(i_1,i_2,i_bin) = bin_2(i_1,i_2,i_bin) + 1.0
```

```
          B_flag = .TRUE.
        END IF
        i_bin = i_bin + 1
      END DO
    END IF

    RETURN

    END


*                    SUBROUTINE binFIL3
*
* Purpose:  Increment density function bin corresponding to input value


    SUBROUTINE BINS_3( i_1, i_2, i_3, Vr, Vr_low, Vr_hgh, N_1, N_2, -
                N_3, bin_3, N_bins )

    REAL*4 bin_3(1:4,1:4,1:24,0:61)
    REAL*4 Vr, Vr_low, Vr_hgh, d_Vr
    INTEGER*4 N_bins, i_1, i_2, i_3, N_1, N_2, N_3

    LOGICAL*1 B_flag

    IF ( Vr .LE. Vr_low ) THEN
        bin_3(i_1,i_2,i_3,0) = bin_3(i_1,i_2,i_3,0) + 1.0
    ELSE IF ( Vr .GT. Vr_hgh ) THEN
          bin_3(i_1,i_2,i_3,N_bins+1) = bin_3(i_1,i_2,i_3,N_bins+1) + 1.0
    ELSE
      i_bin = 1
      B_flag = .FALSE.
        d_Vr = ( Vr_hgh - Vr_low ) / FLOAT( N_bins )
      DO WHILE ( i_bin .LE. N_bins .AND. .NOT. B_flag )
          Vr_min = Vr_low + ( i_bin - 1 ) * d_Vr
        Vr_max = Vr_min + d_Vr
        IF ( Vr .GT. Vr_min .AND. Vr .LE. Vr_max ) THEN
            bin_3(i_1,i_2,i_3,i_bin) = bin_3(i_1,i_2,i_3,i_bin) + 1.0
            B_flag = .TRUE.
        END IF
        i_bin = i_bin + 1
      END DO
    END IF

    RETURN
```

```
      END


C               SUBROUTINE PARMBN2
C
C  Purpose:  Enter parameter values from UAG-23.NEW formatted files into
C         appropriate arrays holding all hourly values for a given month.

      SUBROUTINE PARMBN2( i_SITE, i_DAY, i_HOUR, i_CODE, P_value, Q_char, -
                  D_char )

      REAL*4 P_value
      INTEGER*4 i_SITE, i_DAY, i_HOUR, i_CODE
      CHARACTER*1 Q_char, D_char

      REAL*4 foF2, fxF2, fzF2, M3000F2, hF2, hpf2, foF1, fxF1,-
           M3000F1, hF1, hF, foE, foE2, hE, hE2, foEs,-
           fxEs, fbEs, fEs, hEs, foF1d5, fmin, M3000F1d5, hf1d5,-
           foI, fxI, fmI, I2000, II, Ixxx

C     CHARACTER*1 QfoF2, QfxF2, QfzF2, QM3000F2, QhF2, Qhpf2, QfoF1, QfxF1,-
C                 QM3000F1, QhF1, QhF, QfoE, QfoE2, QhE, QhE2, QfoEs,-
C                 QfxEs, QfbEs, QfEs, QhEs, QfoF1d5, Qfmin, QM3000F1d5,-
C                 Qhf1d5, QfoI, QfxI, QfmI, QI2000, QI, QIxxx

C     CHARACTER*1 DfoF2, DfxF2, DfzF2, DM3000F2, DhF2, Dhpf2, DfoF1, DfxF1,-
C                 DM3000F1, DhF1, DhF, DfoE, DfoE2, DhE, DhE2, DfoEs,-
C                 DfxEs, DfbEs, DfEs, DhEs, DfoF1d5, Dfmin, DM3000F1d5,-
C                 Dhf1d5, DfoI, DfxI, DfmI, DI2000, DI, DIxxx

C ***************************************************************************
C Define common blocks

      COMMON/IONSNP/ foF2(2,31,24), M3000F2(2,31,24), hF2(2,31,24), -
                foF1(2,31,24), M3000F1(2,31,24), hF1(2,31,24), -
                hF(2,31,24), foE(2,31,24), -
                hE(2,31,24), foEs(2,31,24), fbEs(2,31,24), -
                     fEs(2,31,24), hEs(2,31,24)

C                    fmin(2,31,24), fxI(2,31,24)

      IF ( i_CODE .EQ. 1 ) THEN
        foF2(i_SITE,i_DAY,i_HOUR) = P_value
C       QfoF2(i_SITE,i_DAY,i_HOUR) = Q_char
C       DfoF2(i_SITE,i_DAY,i_HOUR) = D_char
      END IF
```

```
"           IF ( i_CODE .EQ. 2 ) THEN
"             fxF2(i_SITE,i_DAY,i_HOUR) = P_value
"             QfxF2(i_SITE,i_DAY,i_HOUR) = Q_char
"             DfxF2(i_SITE,i_DAY,i_HOUR) = D_char
"           END IF

"           IF ( i_CODE .EQ. 3 ) THEN
"             fzF2(i_SITE,i_DAY,i_HOUR) = P_value
"             QfzF2(i_SITE,i_DAY,i_HOUR) = Q_char
"             DfzF2(i_SITE,i_DAY,i_HOUR) = D_char
"           END IF

      IF ( i_CODE .EQ. 4 ) THEN
        M3000F2(i_SITE,i_DAY,i_HOUR) = P_value
"       QM3000F2(i_SITE,i_DAY,i_HOUR) = Q_char
"       DM3000F2(i_SITE,i_DAY,i_HOUR) = D_char
      END IF

      IF ( i_CODE .EQ. 5 ) THEN
        hF2(i_SITE,i_DAY,i_HOUR) = P_value
"       QhF2(i_SITE,i_DAY,i_HOUR) = Q_char
"       DhF2(i_SITE,i_DAY,i_HOUR) = D_char
      END IF

"           IF ( i_CODE .EQ. 6 ) THEN
"             hpF2(i_SITE,i_DAY,i_HOUR) = P_value
"             QhpF2(i_SITE,i_DAY,i_HOUR) = Q_char
"             DhpF2(i_SITE,i_DAY,i_HOUR) = D_char
"           END IF

      IF ( i_CODE .EQ. 7 ) THEN
        foF1(i_SITE,i_DAY,i_HOUR) = P_value
"       QfoF1(i_SITE,i_DAY,i_HOUR) = Q_char
"       DfoF1(i_SITE,i_DAY,i_HOUR) = D_char
      END IF

"           IF ( i_CODE .EQ. 8 ) THEN
"             fxF1(i_SITE,i_DAY,i_HOUR) = P_value
"             QfxF1(i_SITE,i_DAY,i_HOUR) = Q_char
"             DfxF1(i_SITE,i_DAY,i_HOUR) = D_char
"           END IF

      IF ( i_CODE .EQ. 9 ) THEN
        M3000F1(i_SITE,i_DAY,i_HOUR) = P_value
"       QM3000F1(i_SITE,i_DAY,i_HOUR) = Q_char
"       DM3000F1(i_SITE,i_DAY,i_HOUR) = D_char
```

```
      END IF

      IF ( i_CODE .EQ. 10 ) THEN
        hF1(i_SITE,i_DAY,i_HOUR) = P_value
!       QhF1(i_SITE,i_DAY,i_HOUR) = Q_char
!       DhF1(i_SITE,i_DAY,i_HOUR) = D_char
      END IF

      IF ( i_CODE .EQ. 11 ) THEN
        hF(i_SITE,i_DAY,i_HOUR) = P_value
!       QhF(i_SITE,i_DAY,i_HOUR) = Q_char
!       DhF(i_SITE,i_DAY,i_HOUR) = D_char
      END IF

      IF ( i_CODE .EQ. 12 ) THEN
        foE(i_SITE,i_DAY,i_HOUR) = P_value
!       QfoE(i_SITE,i_DAY,i_HOUR) = Q_char
!       DfoE(i_SITE,i_DAY,i_HOUR) = D_char
      END IF

!           IF ( i_CODE .EQ. 13 ) THEN
!             foE2(i_SITE,i_DAY,i_HOUR) = P_value
!             QfoE2(i_SITE,i_DAY,i_HOUR) = Q_char
!             DfoE2(i_SITE,i_DAY,i_HOUR) = D_char
!           END IF

      IF ( i_CODE .EQ. 14 ) THEN
        hE(i_SITE,i_DAY,i_HOUR) = P_value
!       QhE(i_SITE,i_DAY,i_HOUR) = Q_char
!       DhE(i_SITE,i_DAY,i_HOUR) = D_char
      END IF

!           IF ( i_CODE .EQ. 15 ) THEN
!             hE2(i_SITE,i_DAY,i_HOUR) = P_value
!             QhE2(i_SITE,i_DAY,i_HOUR) = Q_char
!             DhE2(i_SITE,i_DAY,i_HOUR) = D_char
!           END IF

      IF ( i_CODE .EQ. 16 ) THEN
        foEs(i_SITE,i_DAY,i_HOUR) = P_value
!       QfoEs(i_SITE,i_DAY,i_HOUR) = Q_char
!       DfoEs(i_SITE,i_DAY,i_HOUR) = D_char
      END IF

!           IF ( i_CODE .EQ. 17 ) THEN
!             fxEs(i_SITE,i_DAY,i_HOUR) = P_value
!             QfxEs(i_SITE,i_DAY,i_HOUR) = Q_char
```

```
"           DfxEs(i_SITE,i_DAY,i_HOUR) = D_char
"         END IF

"         IF ( i_CODE .EQ. 18 ) THEN
"           fbEs(i_SITE,i_DAY,i_HOUR) = P_value
"           QfbEs(i_SITE,i_DAY,i_HOUR) = Q_char
"           DfbEs(i_SITE,i_DAY,i_HOUR) = D_char
"         END IF

      IF ( i_CODE .EQ. 19 ) THEN
        fEs(i_SITE,i_DAY,i_HOUR) = P_value
"       QfEs(i_SITE,i_DAY,i_HOUR) = Q_char
"       DfEs(i_SITE,i_DAY,i_HOUR) = D_char
      END IF

      IF ( i_CODE .EQ. 20 ) THEN
        hEs(i_SITE,i_DAY,i_HOUR) = P_value
"       QhEs(i_SITE,i_DAY,i_HOUR) = Q_char
"       DhEs(i_SITE,i_DAY,i_HOUR) = D_char
      END IF

"         IF ( i_CODE .EQ. 21 ) THEN
"           foF1d5(i_SITE,i_DAY,i_HOUR) = P_value
"           QfoF1d5(i_SITE,i_DAY,i_HOUR) = Q_char
"           DfoF1d5(i_SITE,i_DAY,i_HOUR) = D_char
"         END IF

"         IF ( i_CODE .EQ. 22 ) THEN
"           fmin(i_SITE,i_DAY,i_HOUR) = P_value
"           Qfmin(i_SITE,i_DAY,i_HOUR) = Q_char
"           Dfmin(i_SITE,i_DAY,i_HOUR) = D_char
"         END IF

"         IF ( i_CODE .EQ. 23 ) THEN
"           M3000F1d5(i_SITE,i_DAY,i_HOUR) = P_value
"           QM3000F1d5(i_SITE,i_DAY,i_HOUR) = Q_char
"           DM3000F1d5(i_SITE,i_DAY,i_HOUR) = D_char
"         END IF

"         IF ( i_CODE .EQ. 24 ) THEN
"           hF1d5(i_SITE,i_DAY,i_HOUR) = P_value
"           QhF1d5(i_SITE,i_DAY,i_HOUR) = Q_char
"           DhF1d5(i_SITE,i_DAY,i_HOUR) = D_char
"         END IF

"         IF ( i_CODE .EQ. 25 ) THEN
"           foI(i_SITE,i_DAY,i_HOUR) = P_value
```

```
*           QfoI(i_SITE,i_DAY,i_HOUR) = Q_char
*           DfoI(i_SITE,i_DAY,i_HOUR) = D_char
*         END IF

*         IF ( i_CODE .EQ. 26 ) THEN
*           fxI(i_SITE,i_DAY,i_HOUR) = P_value
*           QfxI(i_SITE,i_DAY,i_HOUR) = Q_char
*           DfxI(i_SITE,i_DAY,i_HOUR) = D_char
*         END IF

*         IF ( i_CODE .EQ. 27 ) THEN
*           fmI(i_SITE,i_DAY,i_HOUR) = P_value
*           QfmI(i_SITE,i_DAY,i_HOUR) = Q_char
*           DfmI(i_SITE,i_DAY,i_HOUR) = D_char
*         END IF

*         IF ( i_CODE .EQ. 28 ) THEN
*           I2000(i_SITE,i_DAY,i_HOUR) = P_value
*           QI2000(i_SITE,i_DAY,i_HOUR) = Q_char
*           DI2000(i_SITE,i_DAY,i_HOUR) = D_char
*         END IF

*         IF ( i_CODE .EQ. 29 ) THEN
*           II(i_SITE,i_DAY,i_HOUR) = P_value
*           QI(i_SITE,i_DAY,i_HOUR) = Q_char
*           DI(i_SITE,i_DAY,i_HOUR) = D_char
*         END IF

*         IF ( i_CODE .EQ. 30 ) THEN
*           Ixxx(i_SITE,i_DAY,i_HOUR) = P_value
*           QIxxx(i_SITE,i_DAY,i_HOUR) = Q_char
*           DIxxx(i_SITE,i_DAY,i_HOUR) = D_char
*         END IF

      RETURN

      END




*         Function ROOT
*
*   Purpose:  Compute the minimum value of Qmin given the functions
*         PrO and dPrOdt and Function PrMAX, which finds the
```

```
*           maximum value of PrO given Q and the time at which
*           this maximum occurs, i.e., t_MAX
*
*

      REAL*4 FUNCTION ROOT( a_LOW_rd, a_HGH_rd, r_km, As_rd, INTV, ITRN )

      REAL*4 theta_diff, r_km(2)

      LOGICAL*1 INTV, ITRN, D_flag

      PARAMETER( ITMAX = 100, EPS = 3.0e-8 )

      ROOT_ERR = 0.00001
      INTV = .TRUE.
      ITRN = .TRUE.
      D_flag = .FALSE.

      A = a_LOW_rd
      FA = theta_diff( A, a_LOW_rd, a_HGH_rd, r_km, As_rd )

      B = a_HGH_rd
      FB = theta_diff( B, a_LOW_rd, a_HGH_rd, r_km, As_rd )

      IF ( FB * FA .GT. 0.0 ) THEN

        INTV = .FALSE.

        IF ( FA .LT. 0.0 .AND. FB .LT. 0.0 ) THEN
           ROOT = a_LOW_rd
           RETURN
        END IF

         IF ( FA .GT. 0.0 .AND. FB .GT. 0.0 ) ROOT = a_HGH_rd

          RETURN

         END IF

         FC = FB

         DO ITER = 1, ITMAX

          IF ( FB * FC .GT. 0.0 ) THEN

            C = A
            FC = FA
            D = B - A
```

```
      E = D

    END IF

    IF ( ABS( FC ) .LT. ABS( FB ) ) THEN

      A = B
      B = C
      C = A

      FA = FB
      FB = FC
      FC = FA

    END IF

    TOL1 = 2.0 * EPS * ABS( B ) + 0.5 * ROOT_ERR

    XM = 0.5 * ( C - B )

    IF ( ABS( XM ) .LE. TOL1 .OR. FB .EQ. 0.0 ) THEN

      IF ( B .LT. a_LOW_rd + ROOT_ERR ) THEN
          ROOT = a_LOW_rd
      ELSE
          ROOT = B
      END IF

      RETURN

    END IF

    IF ( ABS( E ) .GE. TOL1 .AND. ABS( FA ) .GT. ABS( FB ) ) THEN

      S = FB / FA

      IF ( A .EQ. C ) THEN
          P = 2.0 * XM * S
          Q = 1.0 - S
      ELSE
          Q = FA / FC
          R = FB / FC
          P = S * ( 2.0 * XM * Q * ( Q - R )-
           -( B - A ) * ( R - 1.0 ) )
          Q = ( Q - 1.0 ) * ( R - 1.0 ) * ( S - 1.0 )
      END IF
```

```
      IF ( P .GT. 0.0 ) Q = - Q

      P = ABS( P )

      Y = MIN( 3.0 * XM * Q - ABS( TOL1 * Q ), ABS( E * Q ) )
      IF ( 2.0 * P .LT. Y ) THEN
         E = D
         D = P / Q
      ELSE
         D = XM
         E = D
      END IF

    ELSE

      D = XM
      E = D

    END IF

    A = B
    FA = FB

    IF ( ABS( D ) .GT. TOL1 ) THEN
      B = B + D
    ELSE
      B = B + SIGN( TOL1, XM )
   END IF

   FB = theta_diff( B, a_LOW_rd, a_HGH_rd, r_km, As_rd )

END DO

ITRN = .FALSE.
ROOT = B

RETURN

END



REAL*4 FUNCTION theta_diff( a_rd, a_LOW_rd, a_HGH_rd, r_km, As_rd )

REAL*4 DUM, r_km(2), As_rd, Re_km, a_km, theta_rd(2), alfa_rd(2)
```

```
REAL*4 Rearth_km

INTEGER*4 i_HOP

COMMON /MISC/ Re_km, Rearth_km

alfa_rd(1) = a_rd
alfa_rd(2) = As_rd - a_rd

DO i_HOP = 1, 2

  DUM = r_km(i_HOP)**2 + Rearth_km**2
  DUM = DUM - 2.0 * r_km(i_HOP) * Rearth_km * COS( alfa_rd(i_HOP) )
    a_km = SQRT( DUM )

  DUM = r_km(i_HOP)**2 - a_km**2 - Rearth_km**2
  DUM = DUM / ( 2.0 * Rearth_km * a_km )
    IF ( DUM .GT. 1.0 ) DUM = 1.0
    IF ( DUM .LT. -1.0 ) DUM = -1.0
    theta_rd(i_HOP) = ASIN( DUM )

END DO

theta_diff = theta_rd(1) - theta_rd(2)

RETURN

END
```

APPENDIX B

MEASUREMENT-PREDICTION COMPARISON OF EMPIRICAL FREQUENCY AVAILABILITY VALUES

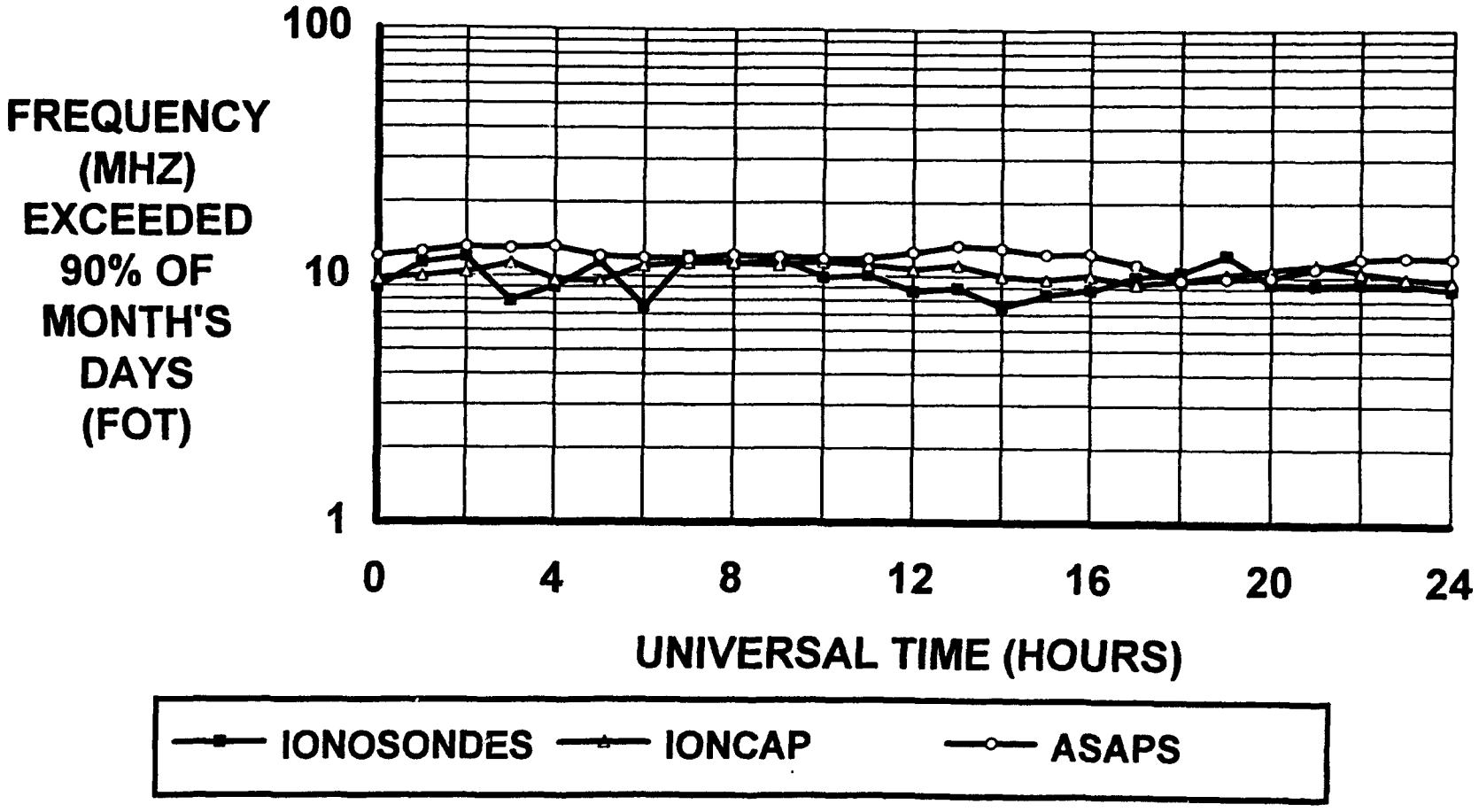
This appendix contains comparison of empirically-derived frequency availability on trans-polar two-hop HF radio links with ASAPS and IONCAP predictions. The empirical data was derived from UAG-23-formatted ionosonde data measured at Dikson, Cape Schmidt, and Loparskaya in Russia and Resolute Bay in Canada in 1991. The 1991 year corresponded to mid-range sunspot numbers from 133 to 172. The following table provides a description of the plots as well as an index to locate specific plots.

TABLE B.1 INDEX TO FREQUENCY AVAILABILITY PLOTS

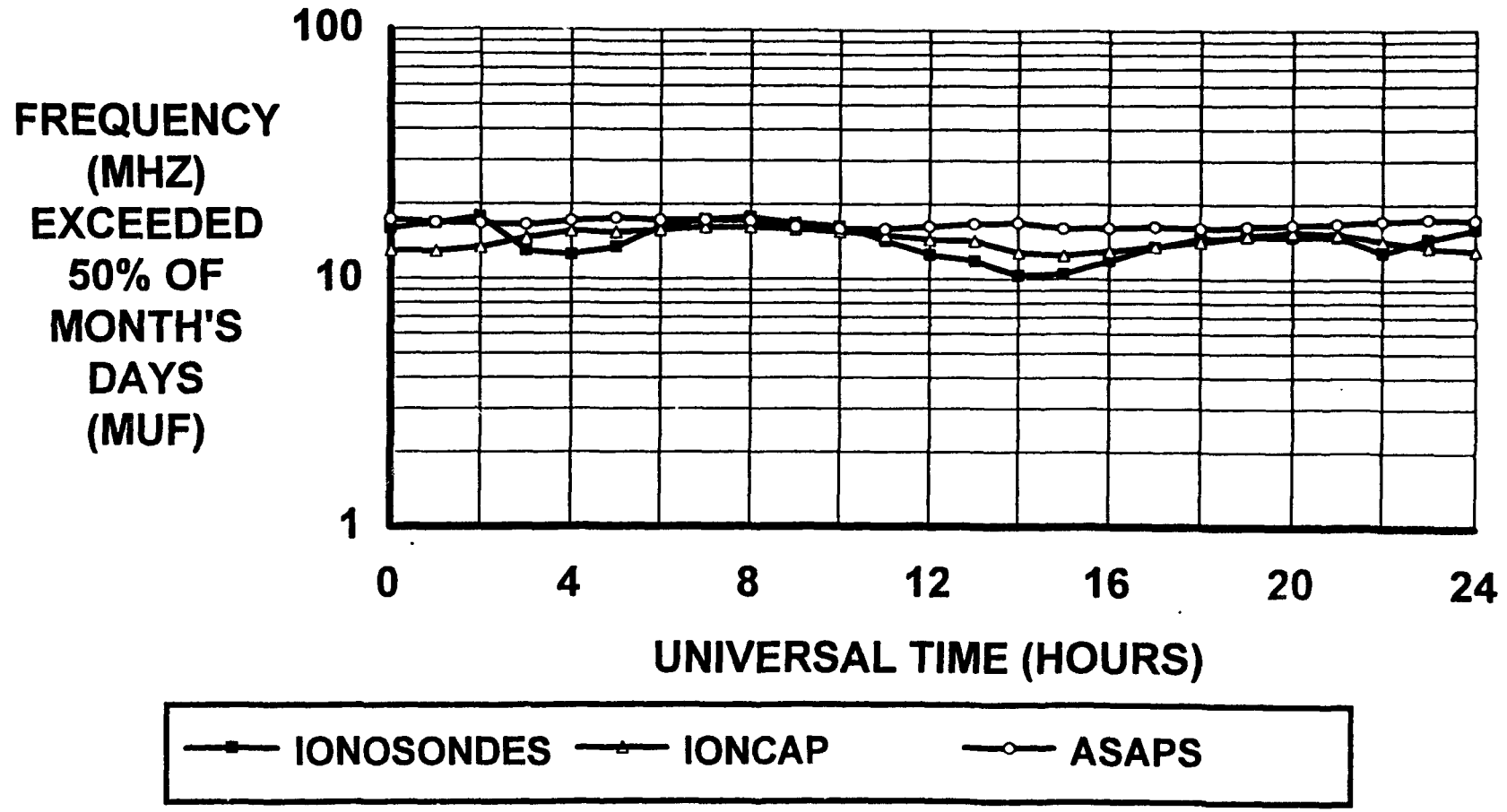
		FOT		MUF		HPF	
LINK	RANGE	MONTH	PAGE	MONTH	PAGE	MONTH	PAGE
DIKSON TO RESOLUTE BAY	7100 KM	JAN	B-3	JAN	B-4	JAN	B-5
		FEB	B-6	FEB	B-7	FEB	B-8
		MAR	B-9	MAR	B-10	MAR	B-11
		APR	B-12	APR	B-13	APR	B-14
		MAY	B-15	MAY	B-16	MAY	B-17
		JUN	B-18	JUN	B-19	JUN	B-20
CAPE SCHMIDT TO LOPARSKAYA	9100 KM	MAR	B-22	MAR	B-23	MAR	B-24
		JUL	B-25	JUL	B-26	JUL	B-27
		SEP	B-28	SEP	B-29	SEP	B-30
		DEC	B-31	DEC	B-32	DEC	B-33

TABLE B.2 INDEX TO FREQUENCY AVAILABILITY PLOTS

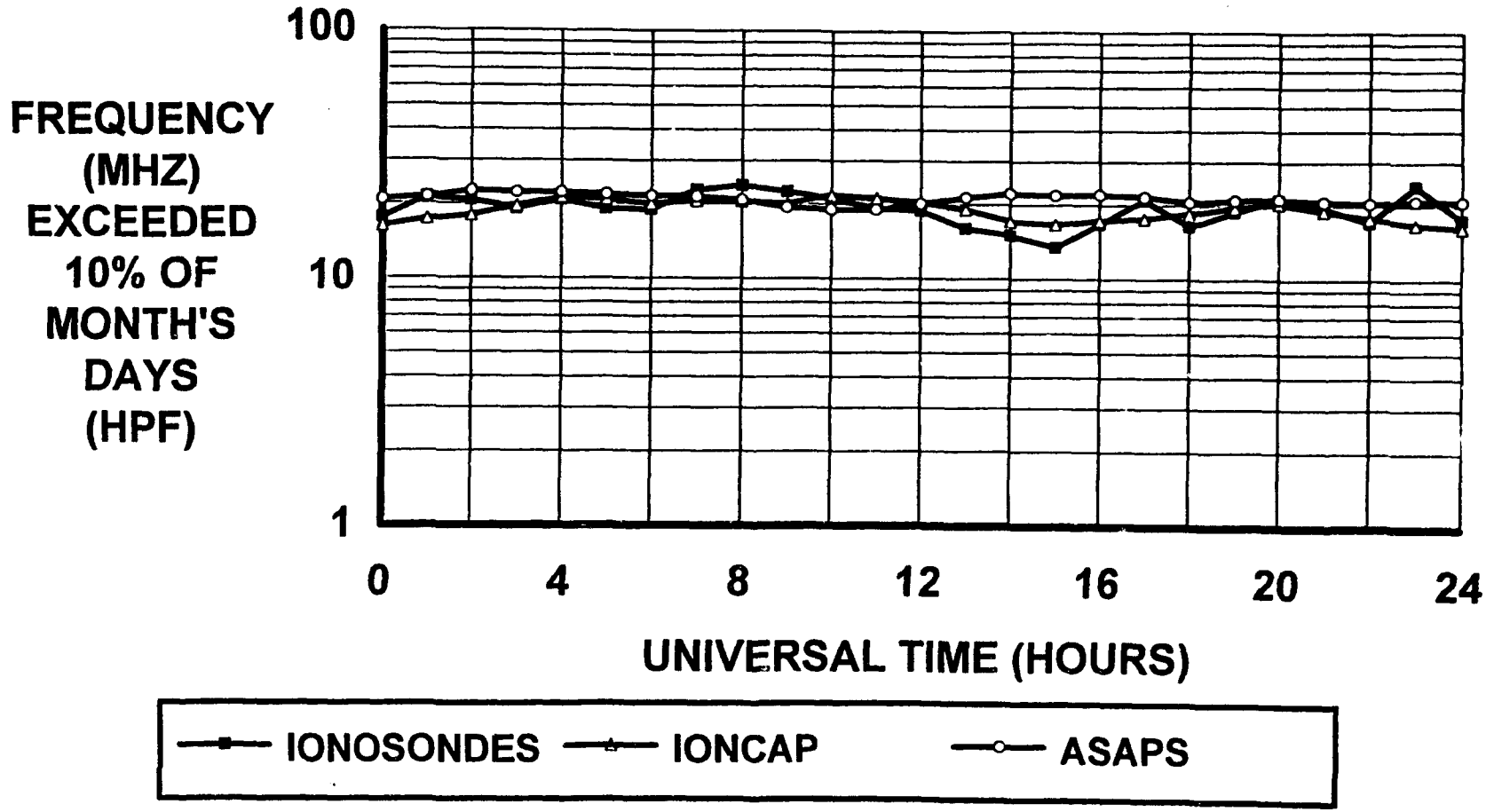
		FOT		MUF		HPF	
LINK	RANGE	MONTH	PAGE	MONTH	PAGE	MONTH	PAGE
[illegible]	[illegible]	JAN	B-3	JAN	B-4	JAN	B-5
[illegible]	[illegible]	FEB	B-6	FEB	B-7	FEB	B-8
[illegible]	9100 KM	MAR	B-9	MAR	B-10	MAR	B-11
[illegible]	[illegible]	APR	B-12	APR	B-13	APR	B-14
[illegible]	[illegible]	MAY	B-15	MAY	B-16	MAY	B-17
[illegible]	[illegible]	JUN	B-18	JUN	B-19	JUN	B-20
CAPE		MAR	B-22	MAR	B-23	MAR	B-24
SCHMIDT	9100 KM	JUL	B-25	JUL	B-26	JUL	B-27
TO		SEP	B-28	SEP	B-29	SEP	B-30
LOPARSKAYA		DEC	B-31	DEC	B-32	DEC	B-33


FOT COMPARISON JAN 1991 SSN: 152 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY
FREQUENCY (MHZ) EXCEEDED 90% OF MONTH'S DAYS (FOT)
100
10
1
0
4
8
12
16
20
24
UNIVERSAL TIME (HOURS)
IONOSONDES
IONCAP
ASAPS

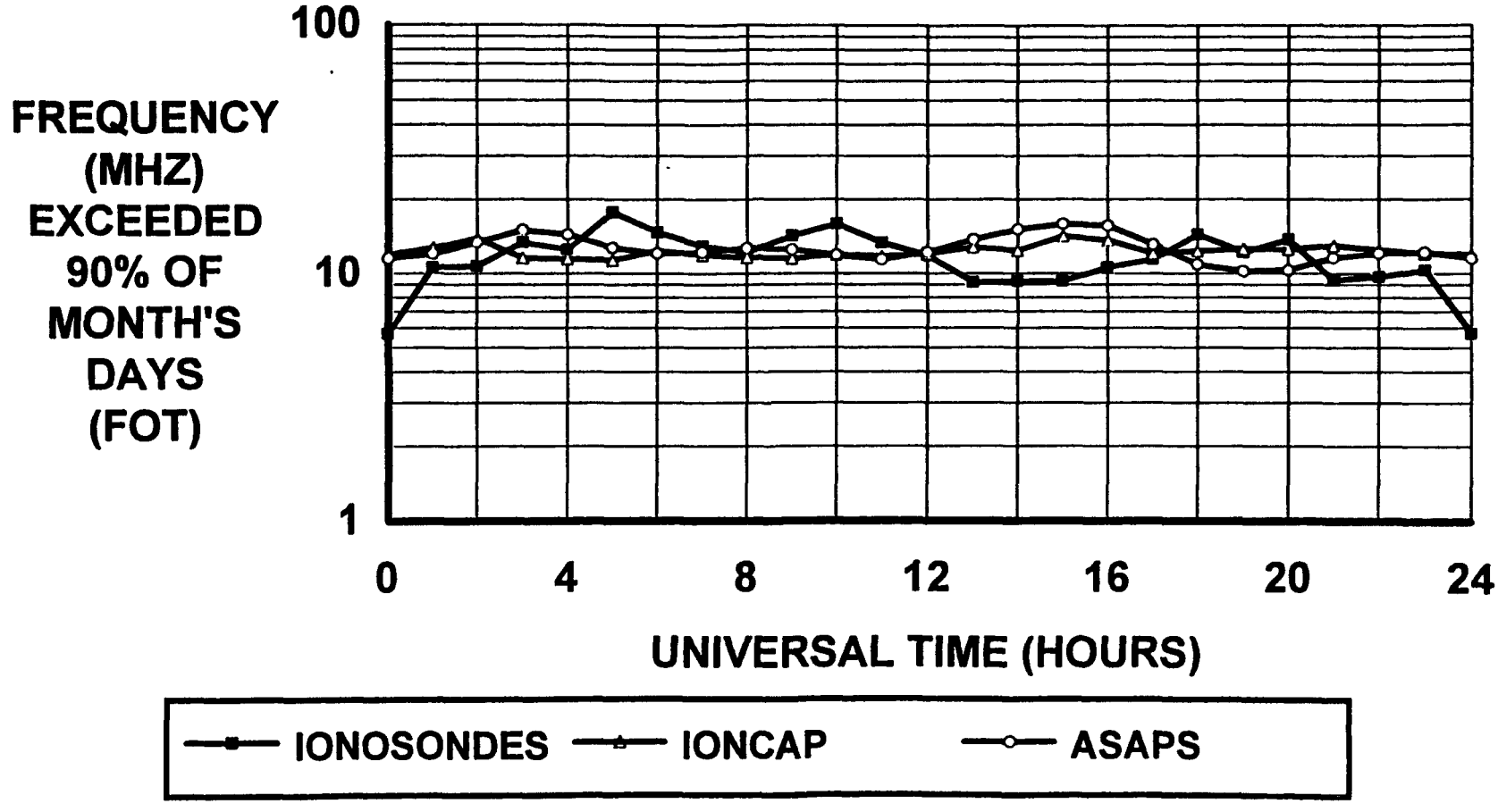
MUF COMPARISON JAN 1991 SSN: 152 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY

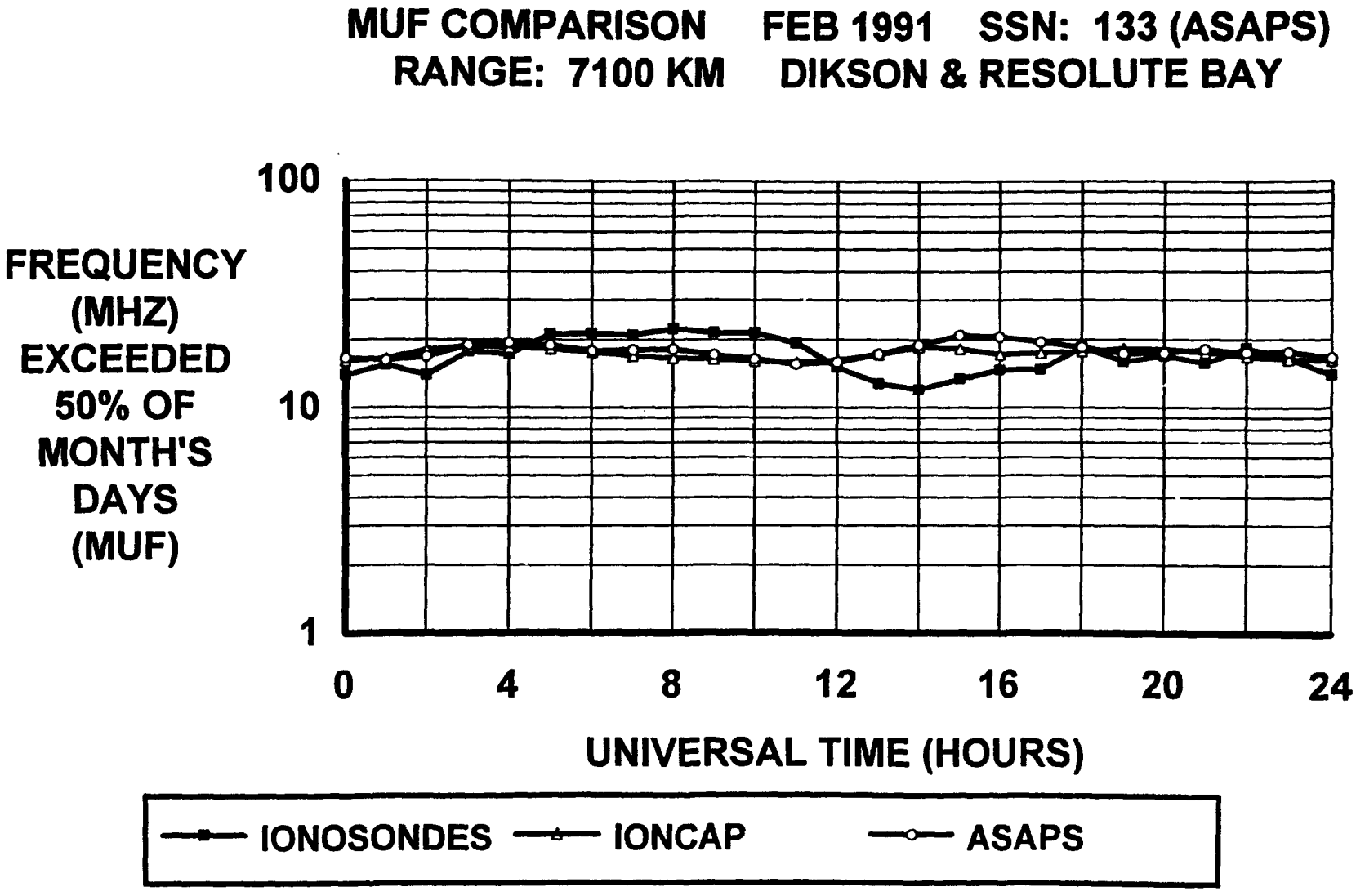


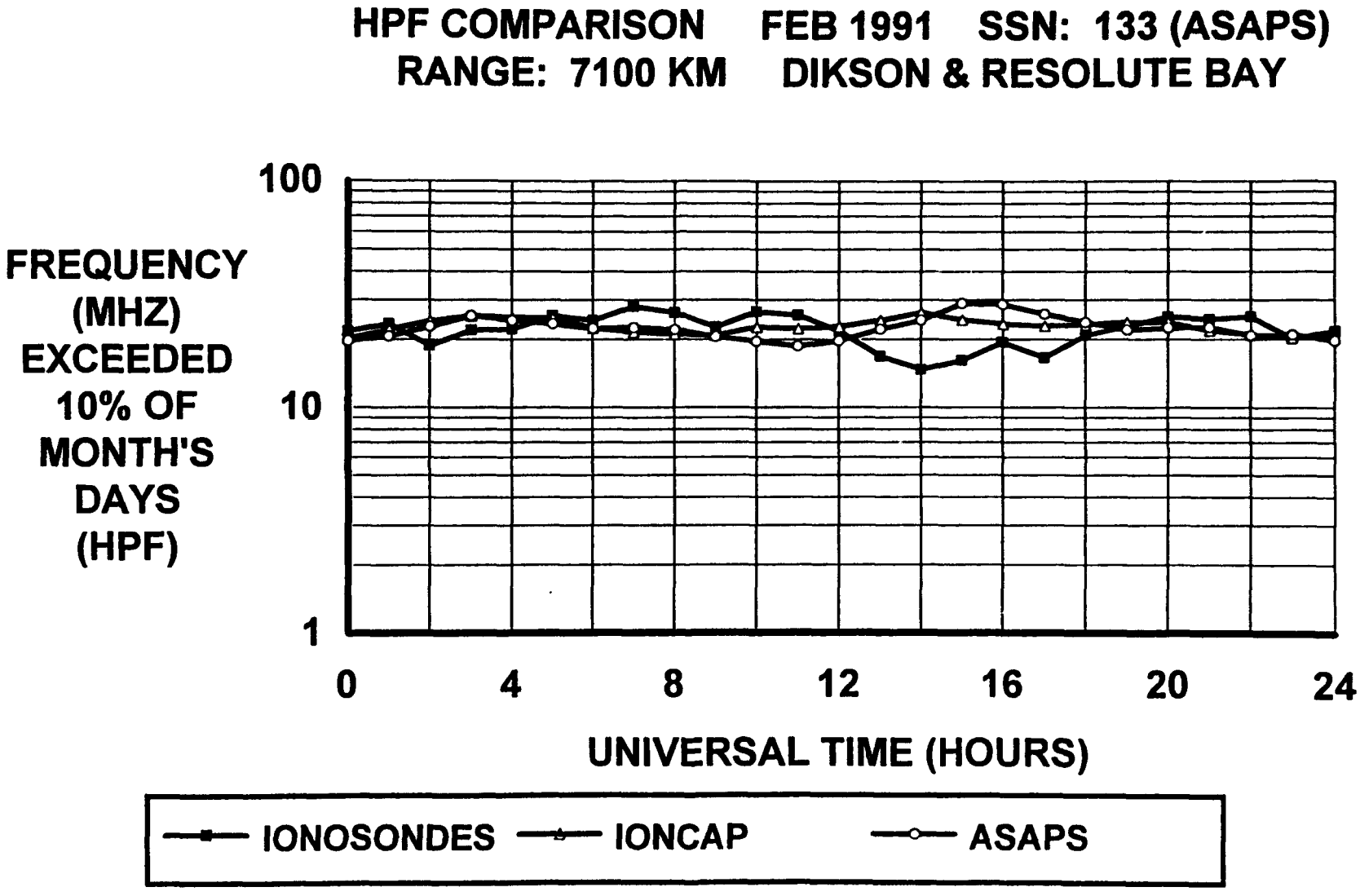
HPF COMPARISON JAN 1991 SSN: 152 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY



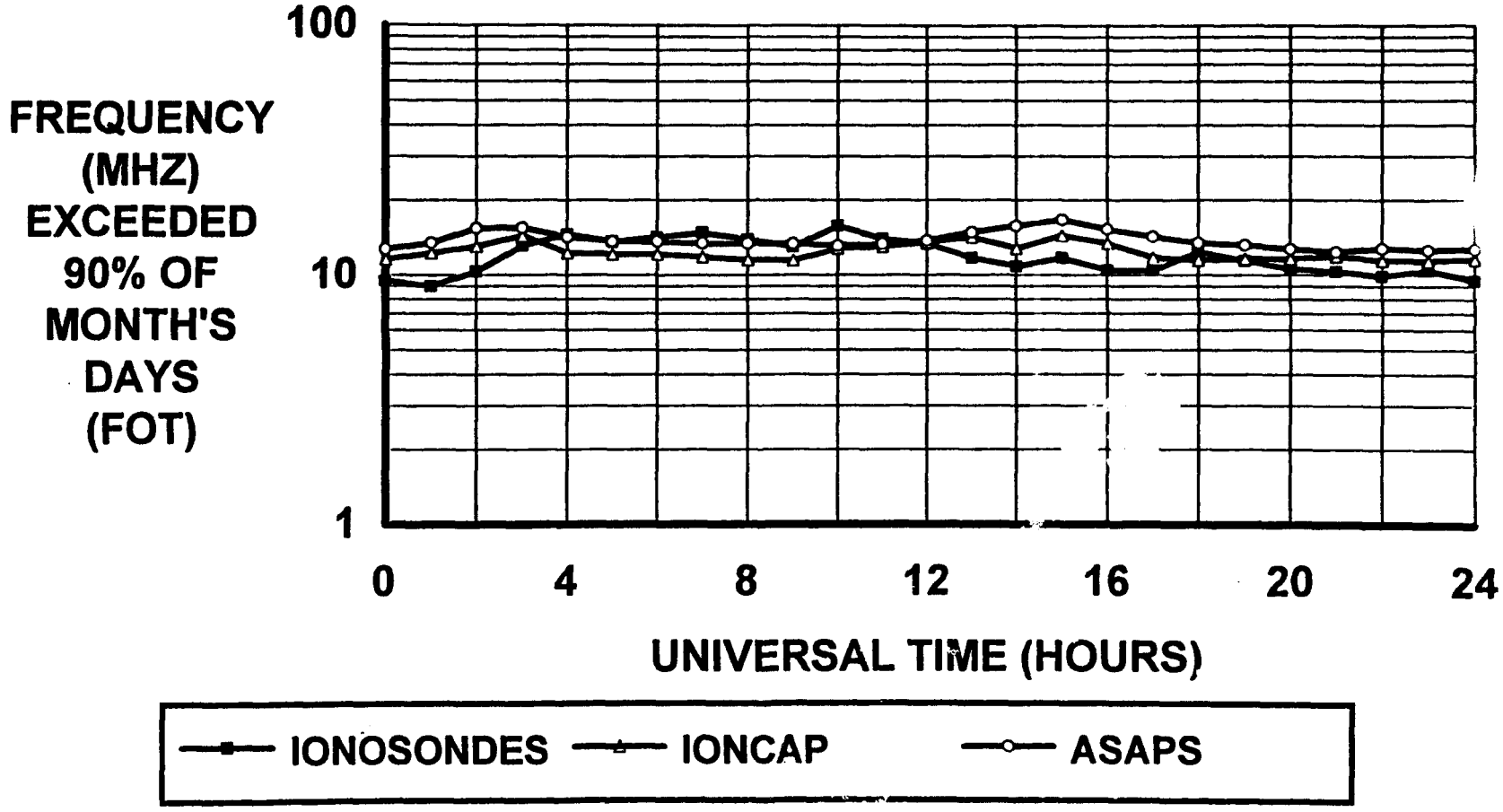
FOT COMPARISON FEB 1991 SSN: 133 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY

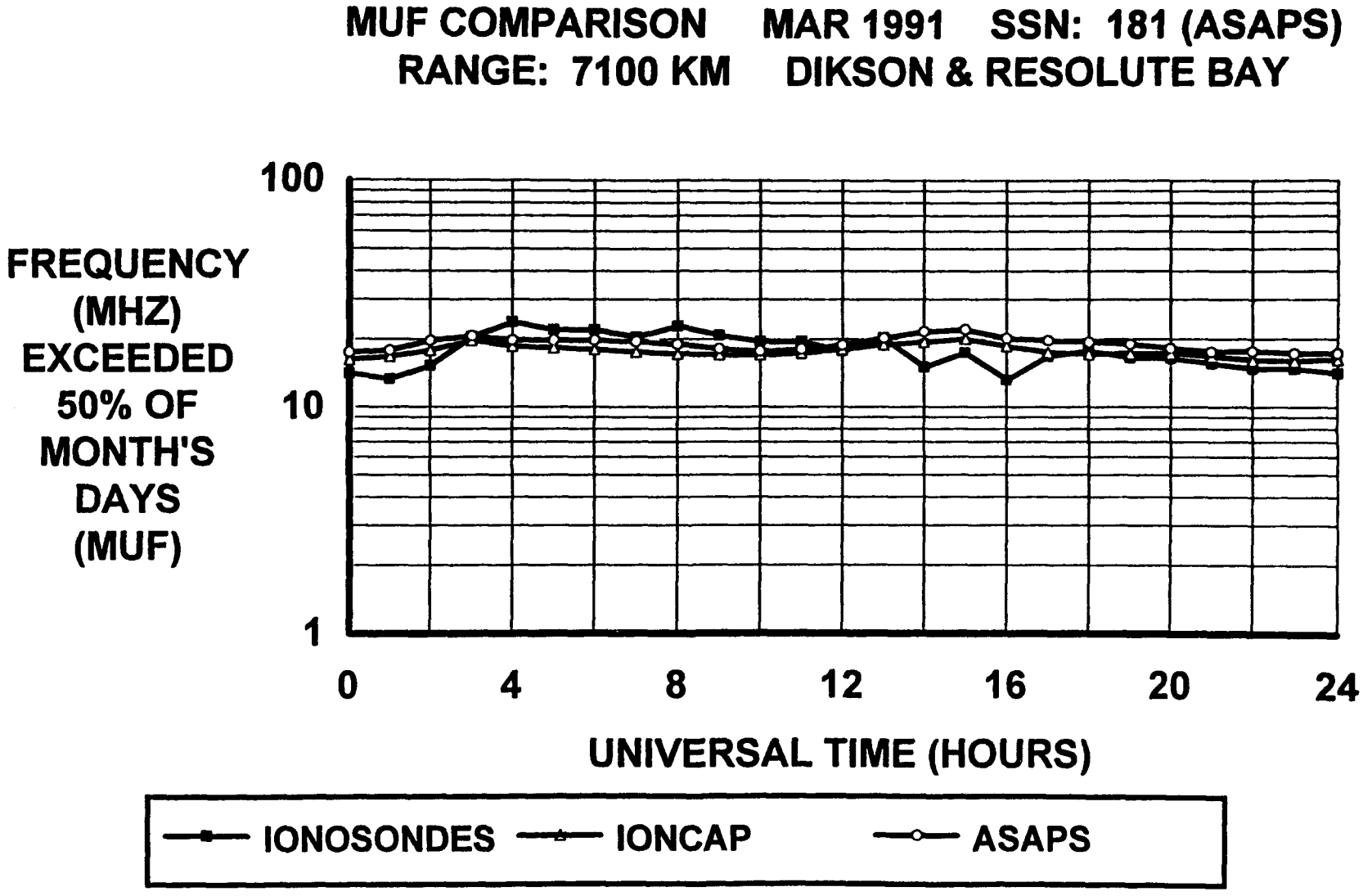


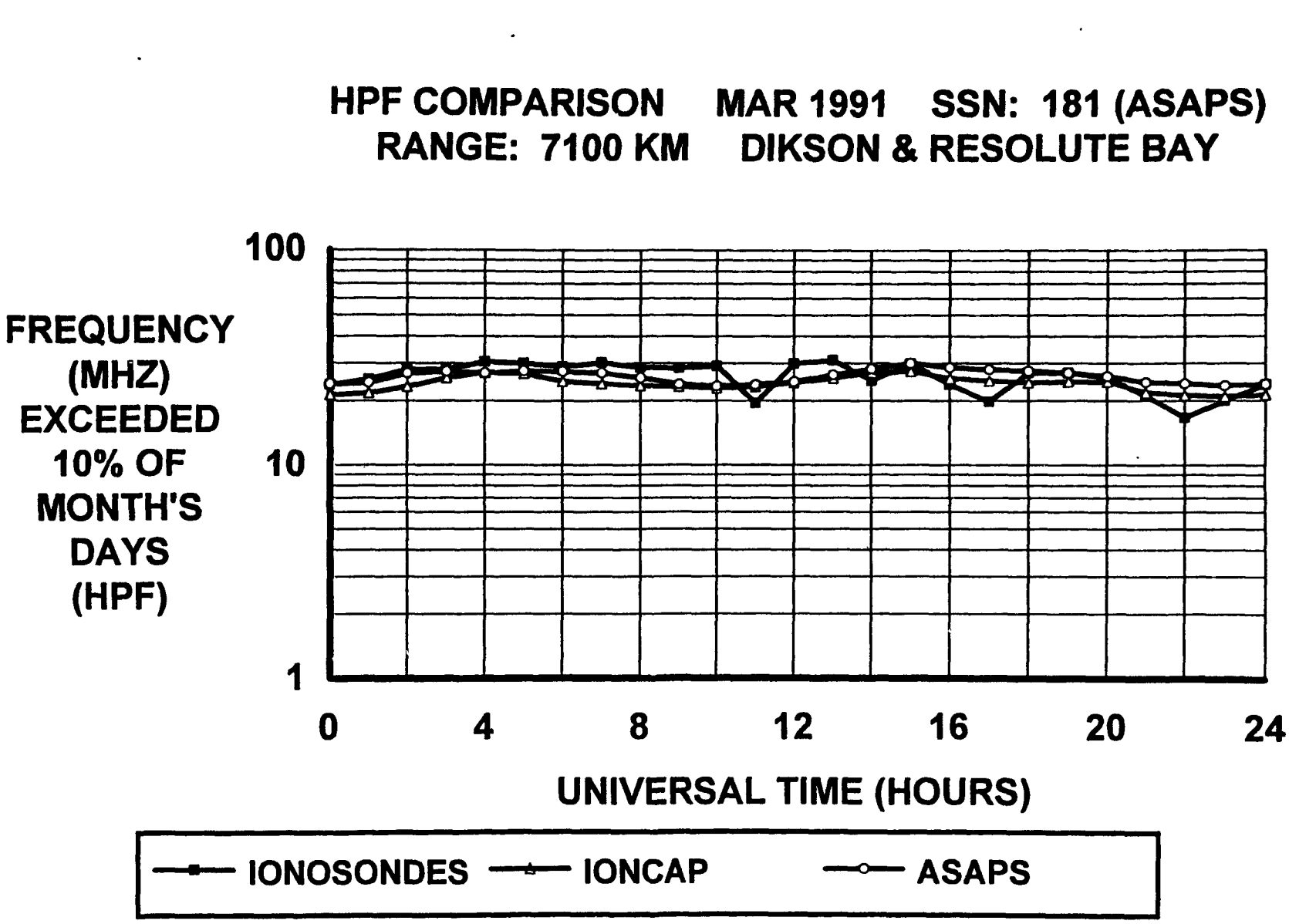

MUF COMPARISON FEB 1991 SSN: 133 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY
FREQUENCY (MHZ) EXCEEDED 50% OF MONTH'S DAYS (MUF)
100
10
1
0
4
8
12
16
20
24
UNIVERSAL TIME (HOURS)
IONOSONDES
IONCAP
ASAPS


HPF COMPARISON FEB 1991 SSN: 133 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY
FREQUENCY (MHZ) EXCEEDED 10% OF MONTH'S DAYS (HPF)
100
10
1
0
4
8
12
16
20
24
UNIVERSAL TIME (HOURS)
IONOSONDES
IONCAP
ASAPS

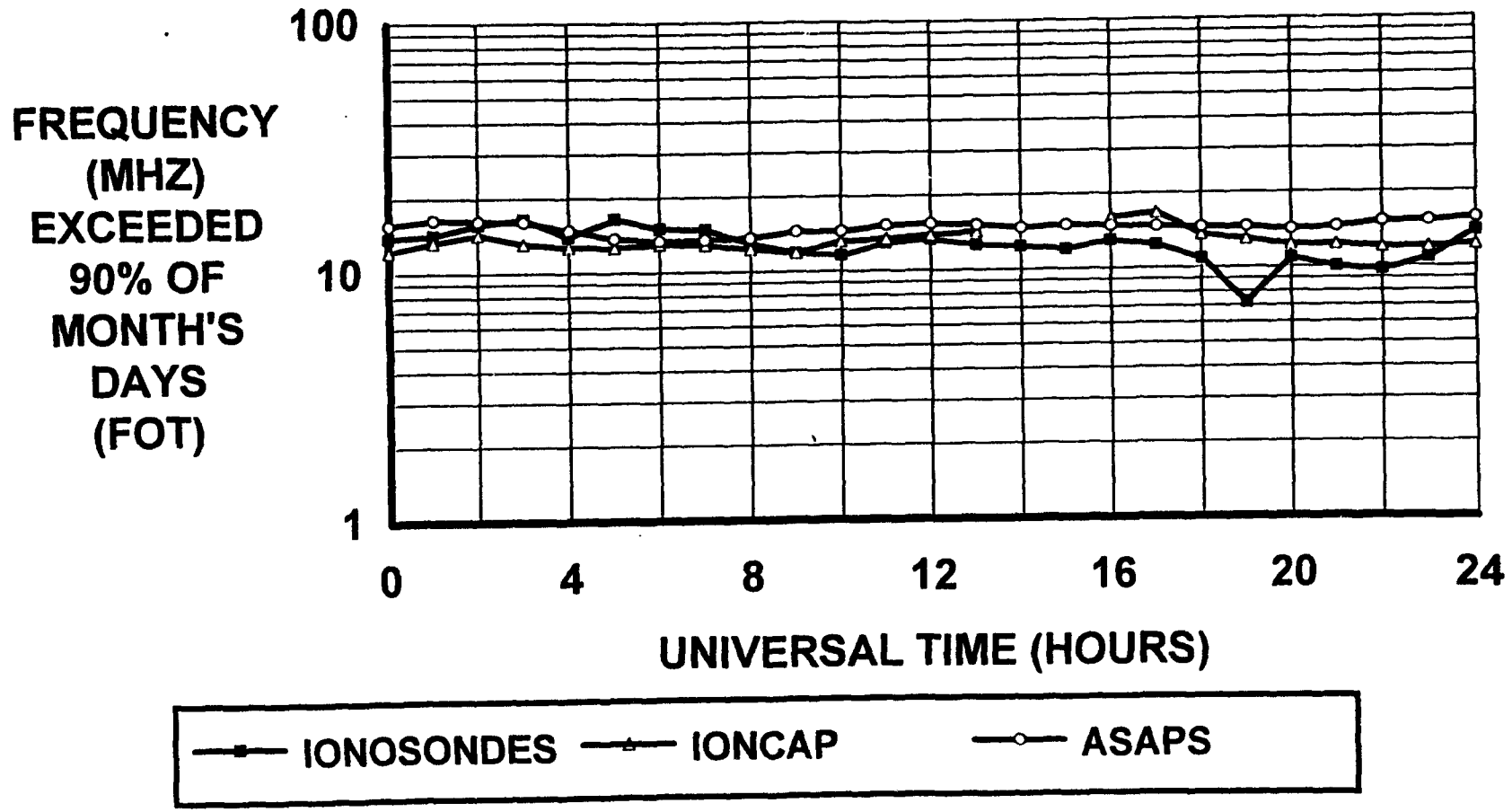
FOT COMPARISON MAR 1991 SSN: 181 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY



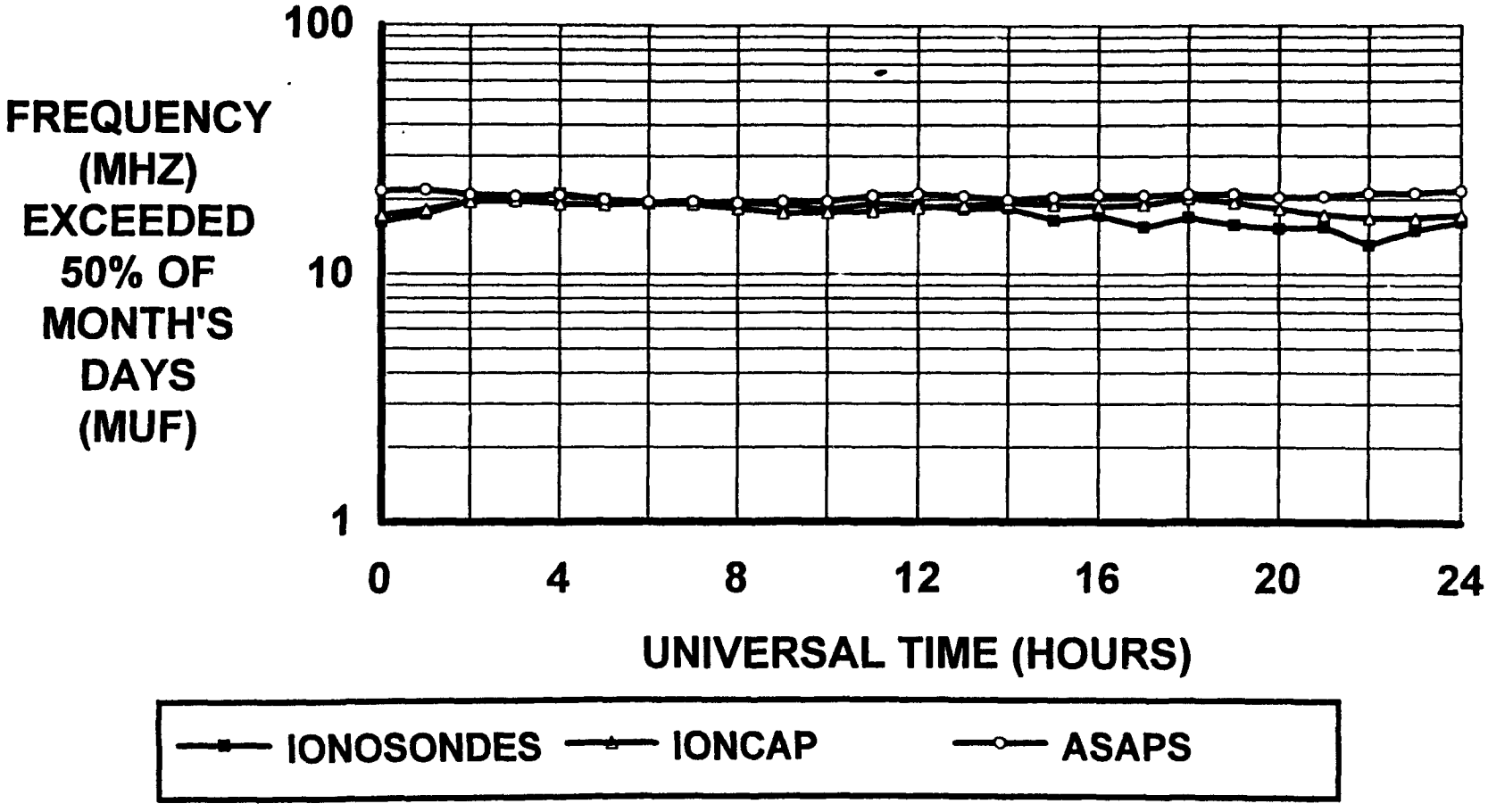

MUF COMPARISON MAR 1991 SSN: 181 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY
FREQUENCY (MHZ) EXCEEDED 50% OF MONTH'S DAYS (MUF)
100
10
1
0
4
8
12
16
20
24
UNIVERSAL TIME (HOURS)
IONOSONDES
IONCAP
ASAPS


HPF COMPARISON MAR 1991 SSN: 181 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY
FREQUENCY (MHZ) EXCEEDED 10% OF MONTH'S DAYS (HPF)
100
10
1
0
4
8
12
16
20
24
UNIVERSAL TIME (HOURS)
IONOSONDES
IONCAP
ASAPS

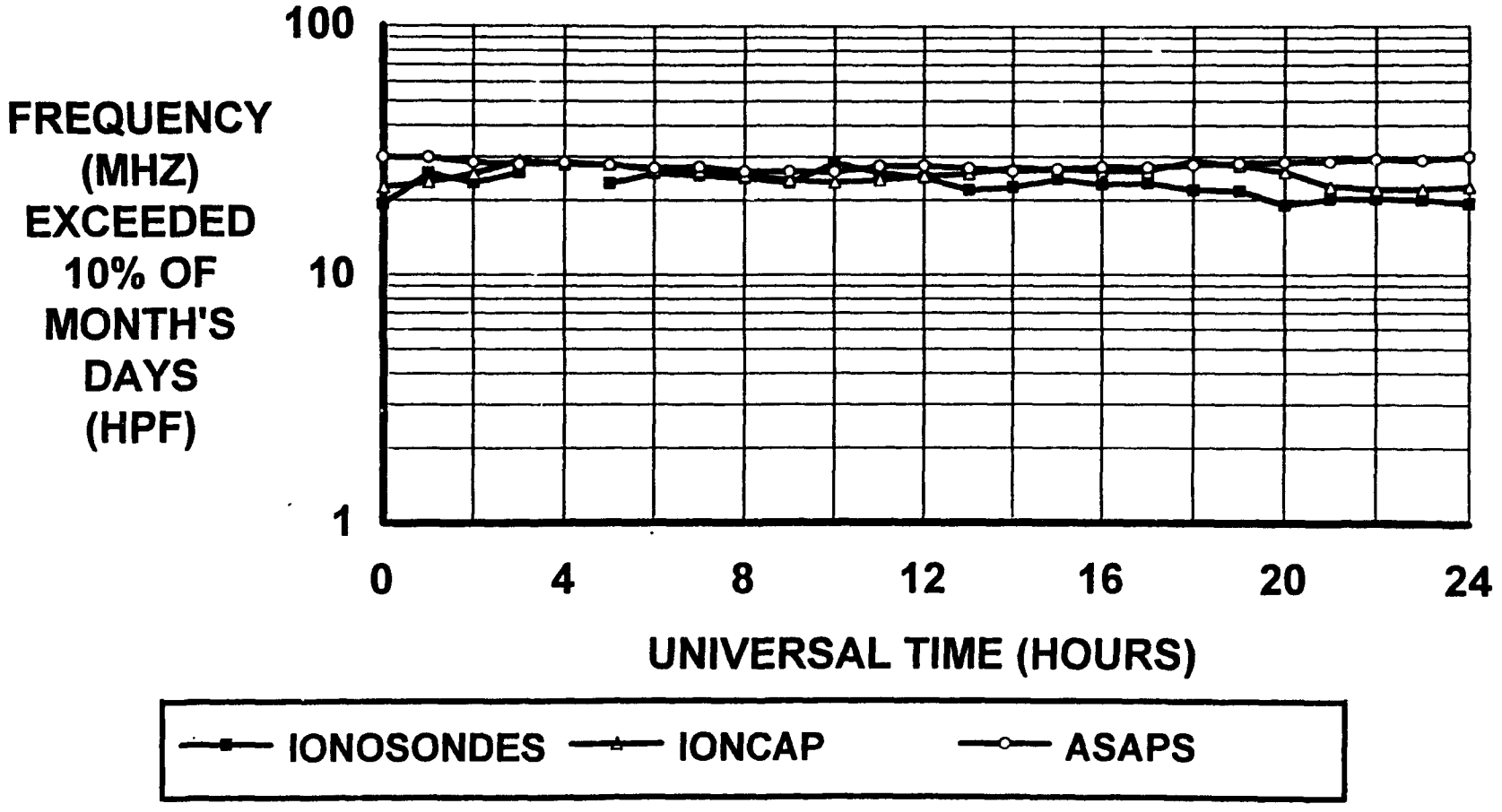
FOT COMPARISON APR 1991 SSN: 172 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY



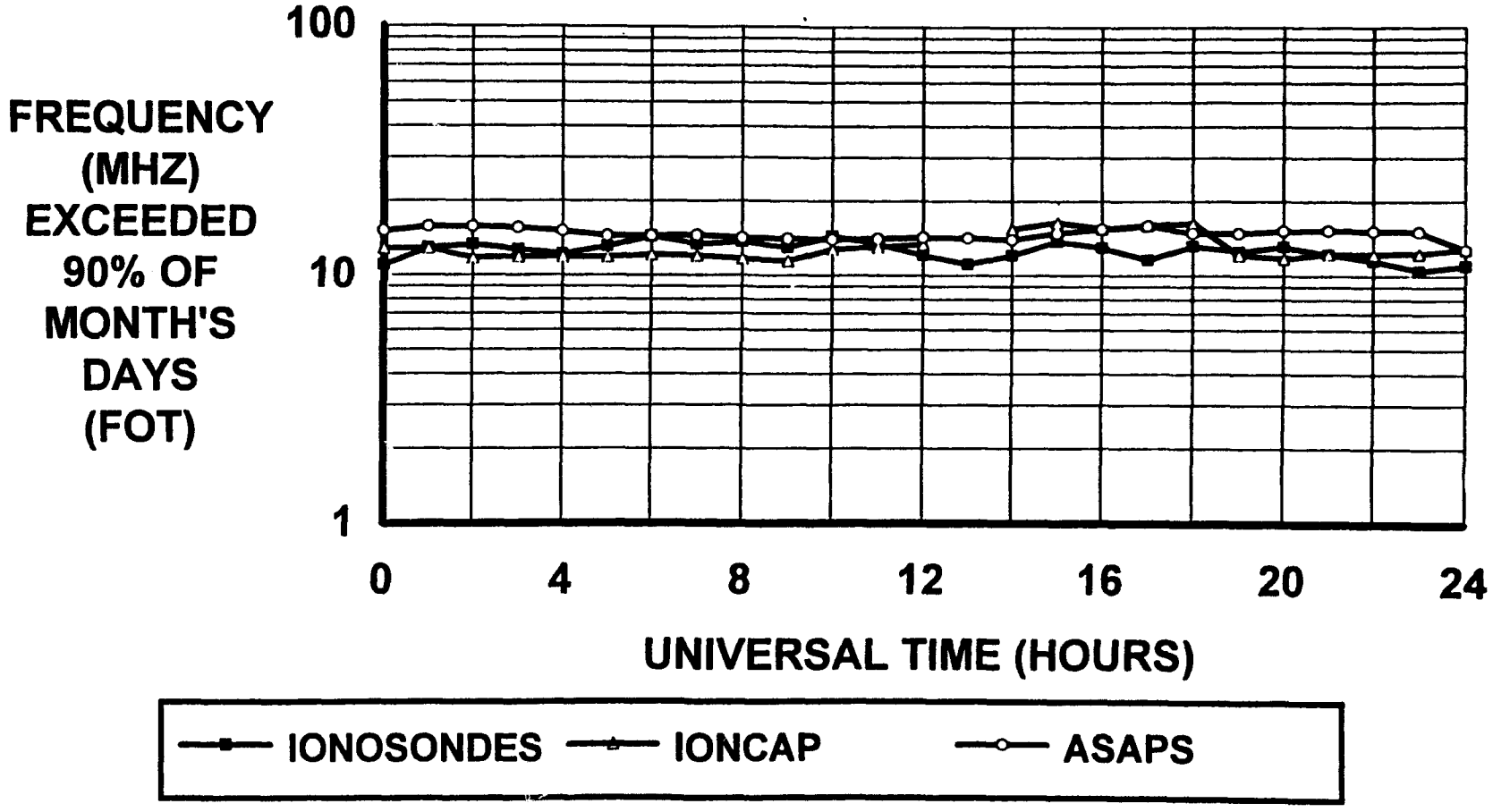
MUF COMPARISON APR 1991 SSN: 172 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY



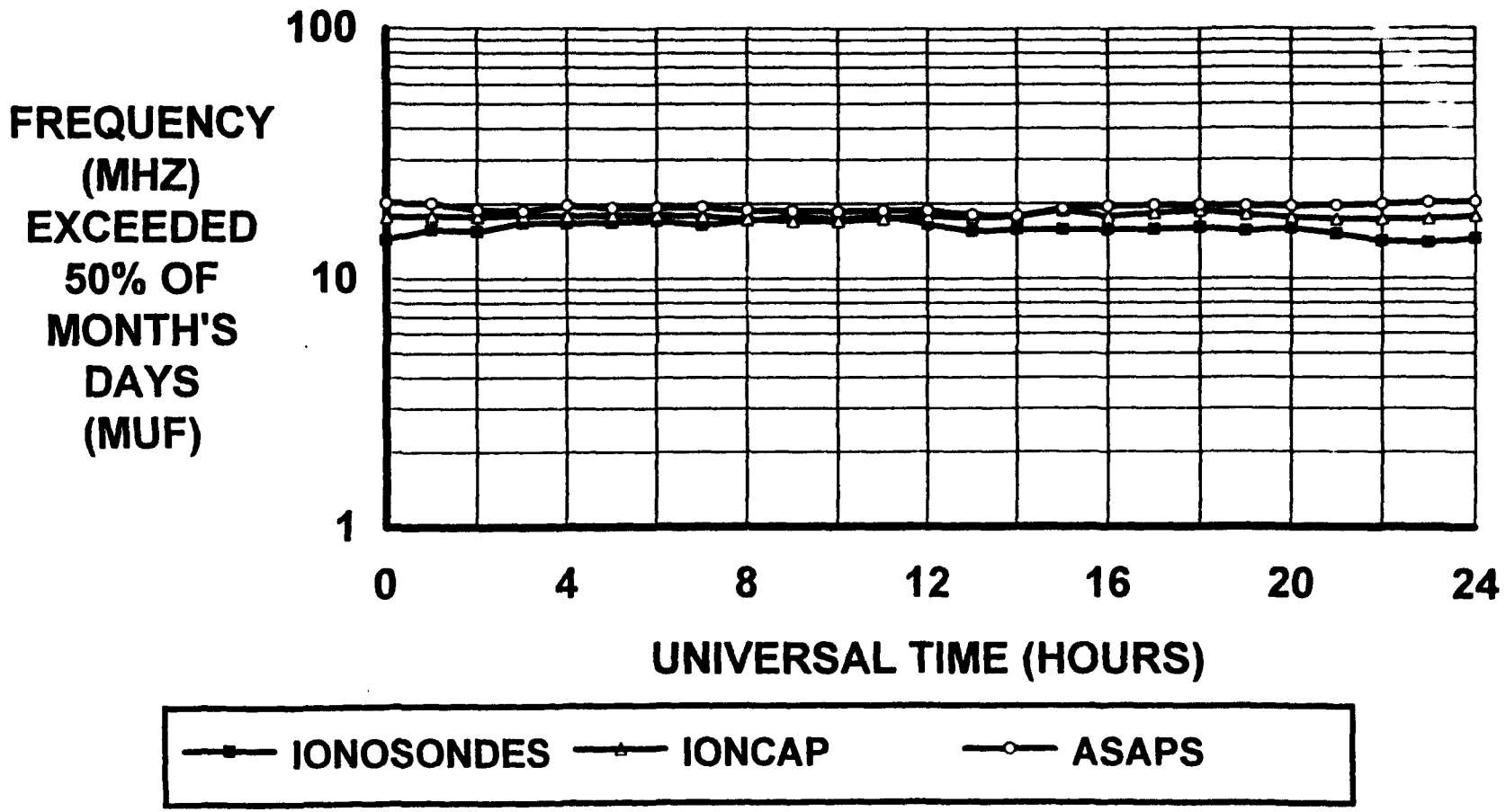
HPF COMPARISON APR 1991 SSN: 172 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY

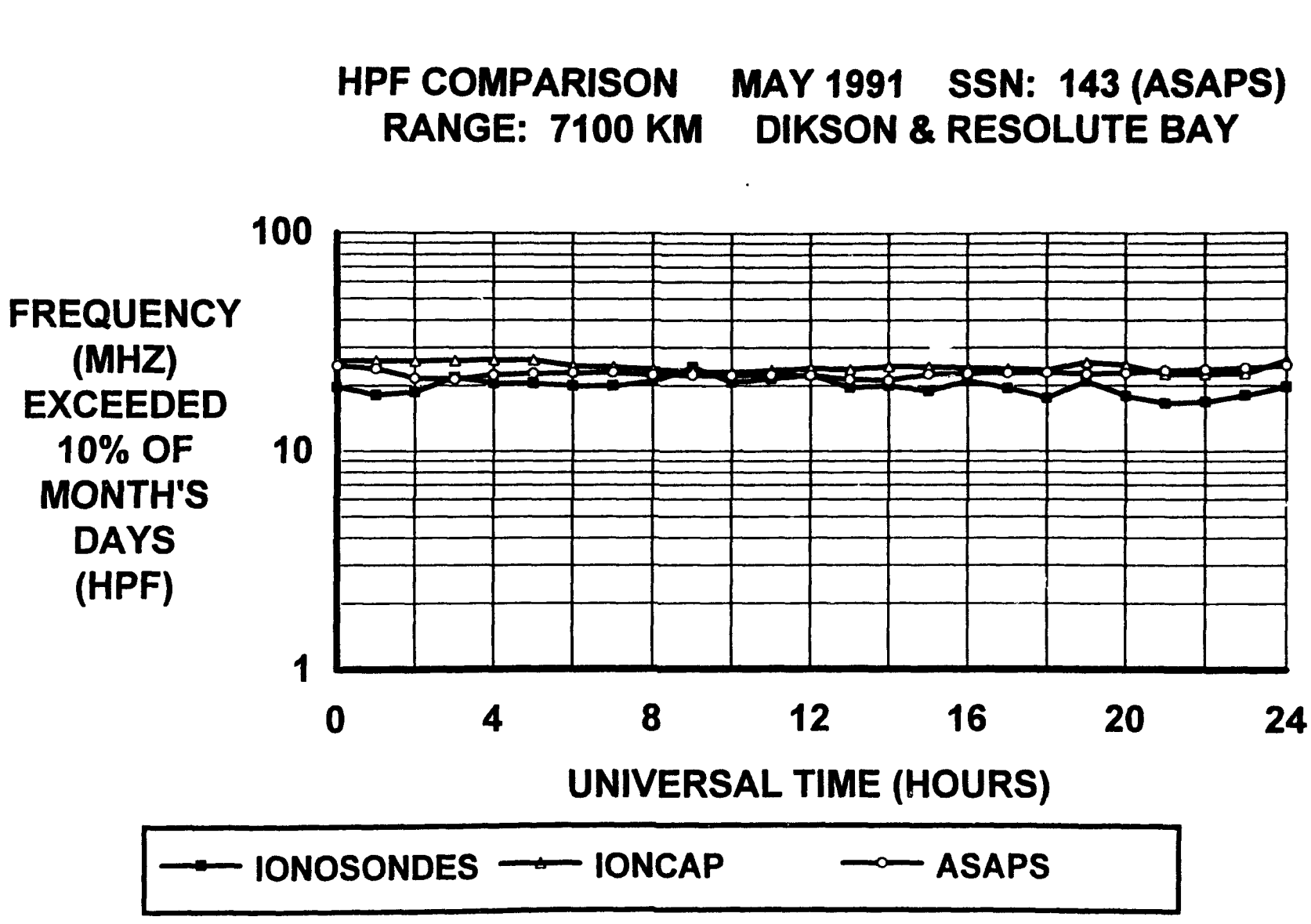


FOT COMPARISON MAY 1991 SSN: 143 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY

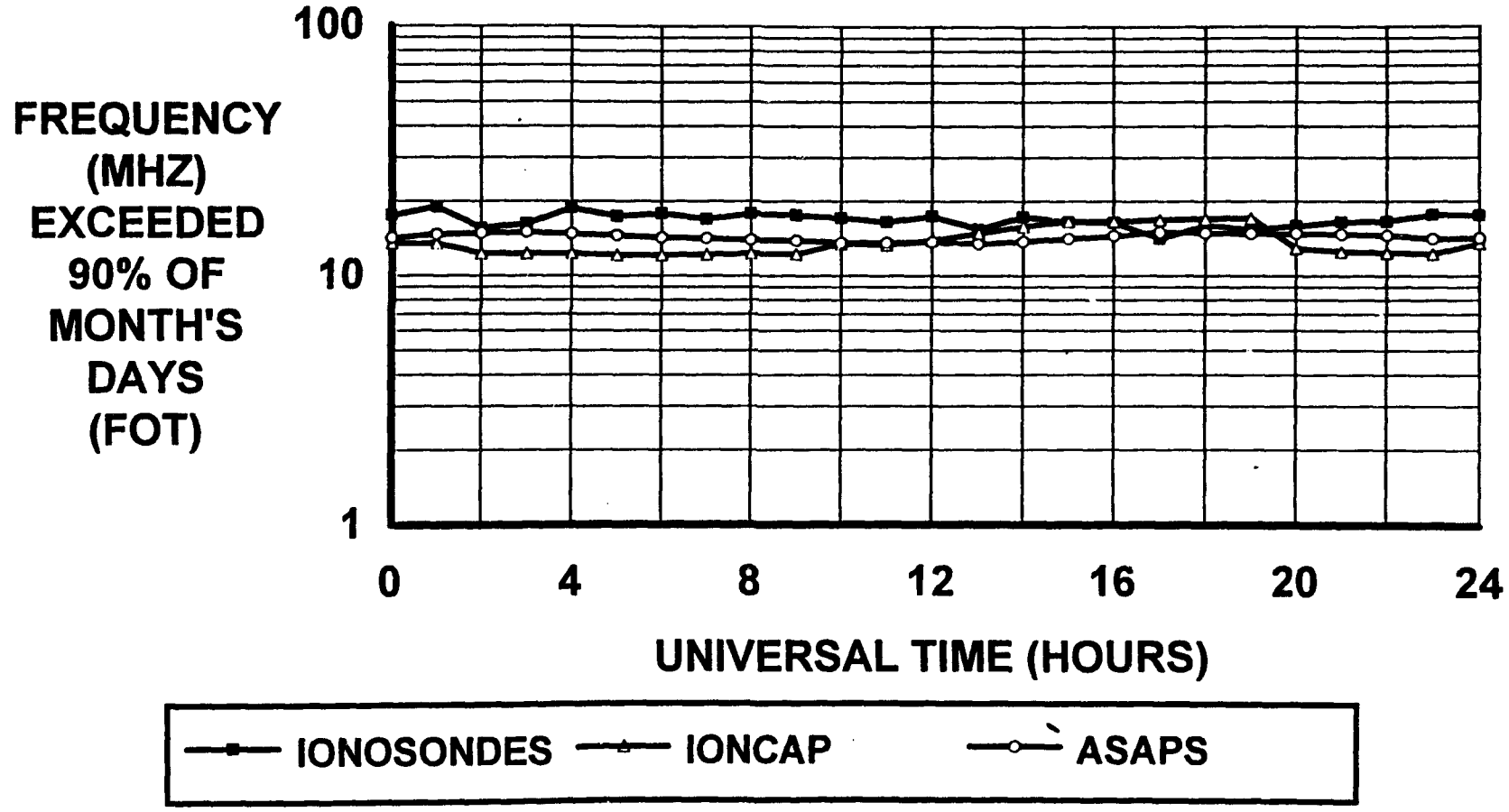


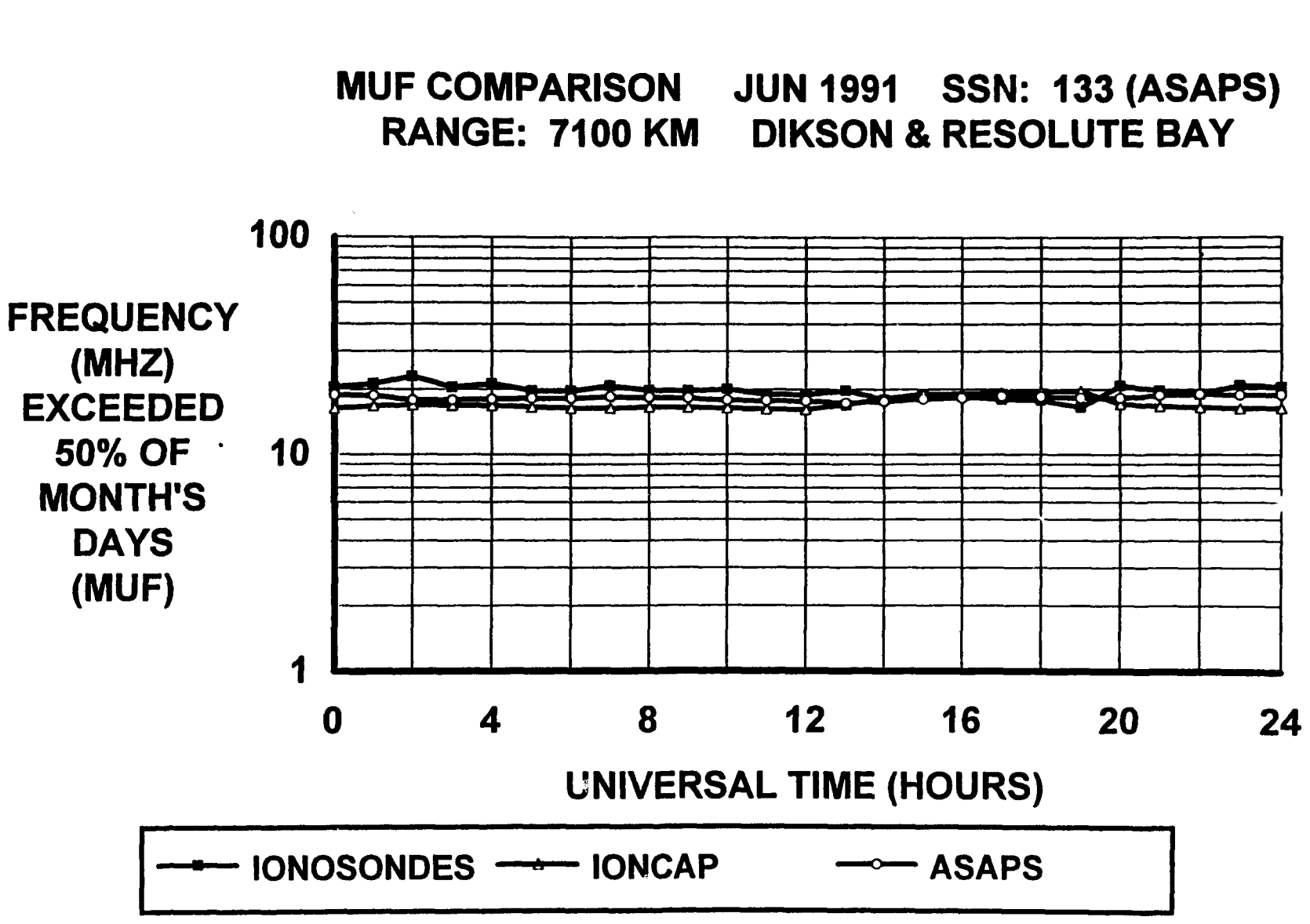
MUF COMPARISON MAY 1991 SSN: 143 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY




HPF COMPARISON MAY 1991 SSN: 143 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY
FREQUENCY (MHZ) EXCEEDED 10% OF MONTH'S DAYS (HPF)
100
10
1
0
4
8
12
16
20
24
UNIVERSAL TIME (HOURS)
IONOSONDES
IONCAP
ASAPS

FOT COMPARISON JUN 1991 SSN: 133 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY




MUF COMPARISON JUN 1991 SSN: 133 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY
FREQUENCY (MHZ) EXCEEDED 50% OF MONTH'S DAYS (MUF)
100
10
1
0
4
8
12
16
20
24
UNIVERSAL TIME (HOURS)
IONOSONDES
IONCAP
ASAPS

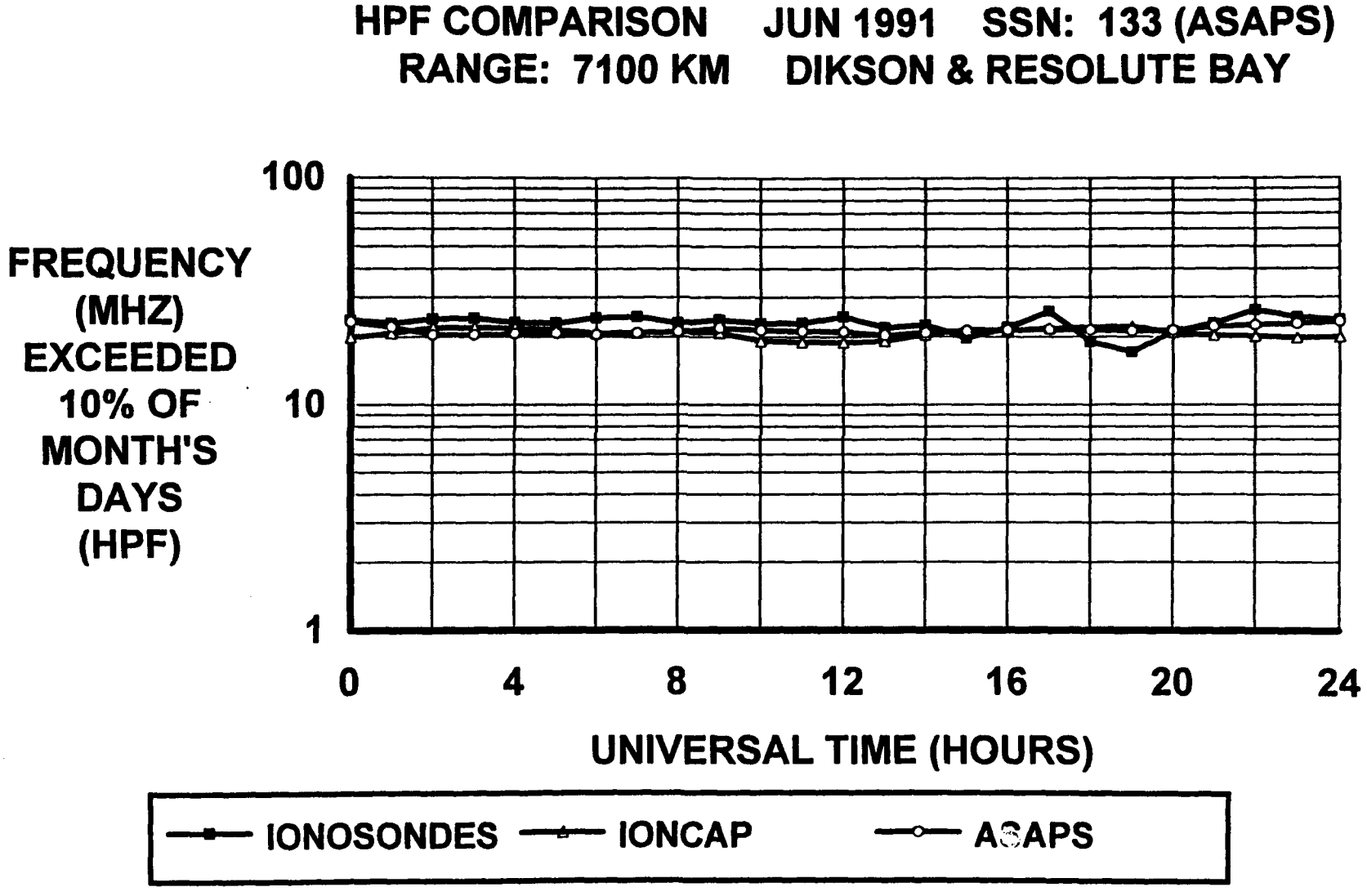
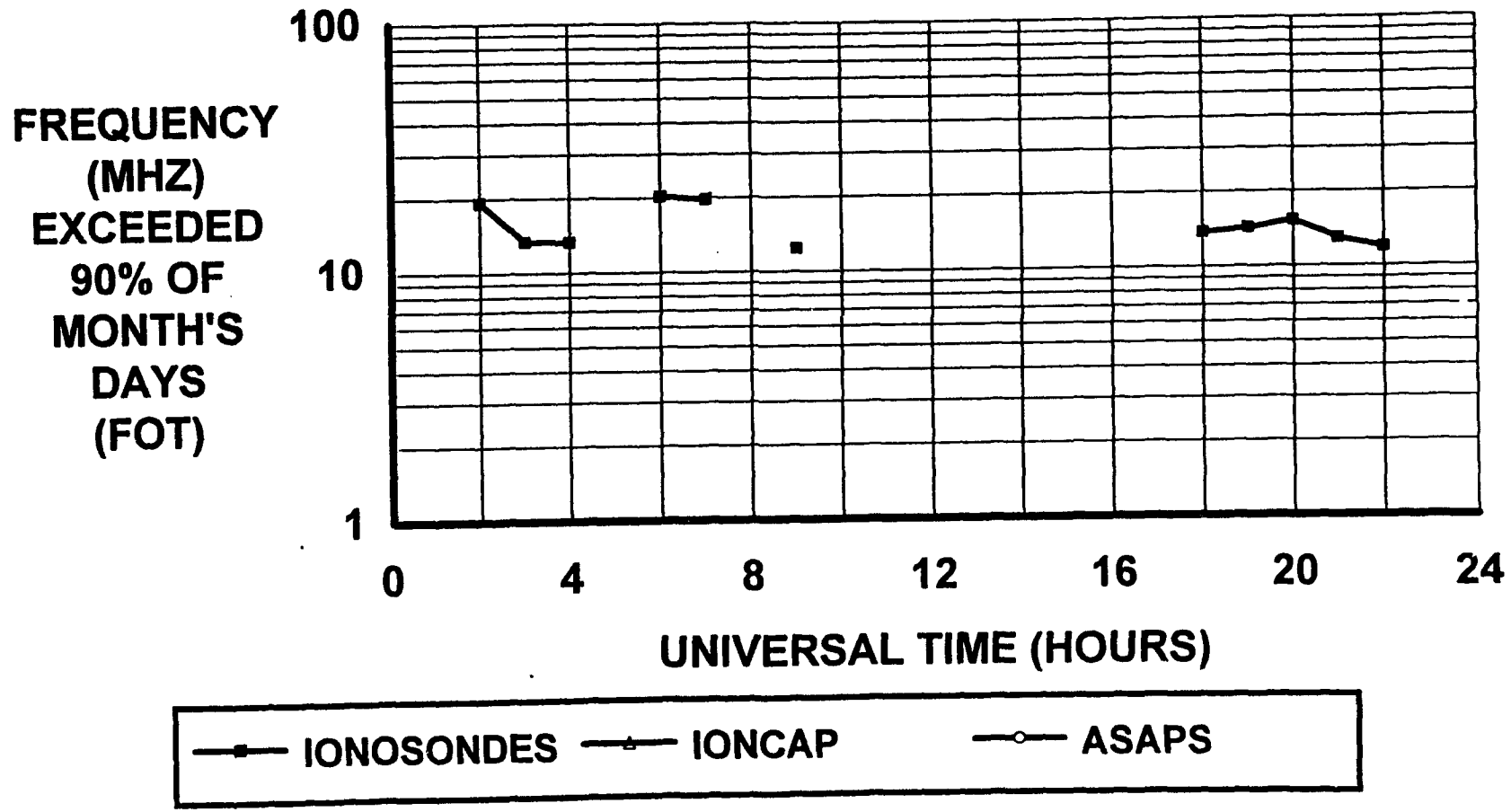

HPF COMPARISON JUN 1991 SSN: 133 (ASAPS)
RANGE: 7100 KM DIKSON & RESOLUTE BAY
FREQUENCY (MHZ) EXCEEDED 10% OF MONTH'S DAYS (HPF)
100
10
1
0
4
8
12
16
20
24
UNIVERSAL TIME (HOURS)
IONOSONDES
IONCAP
ASAPS

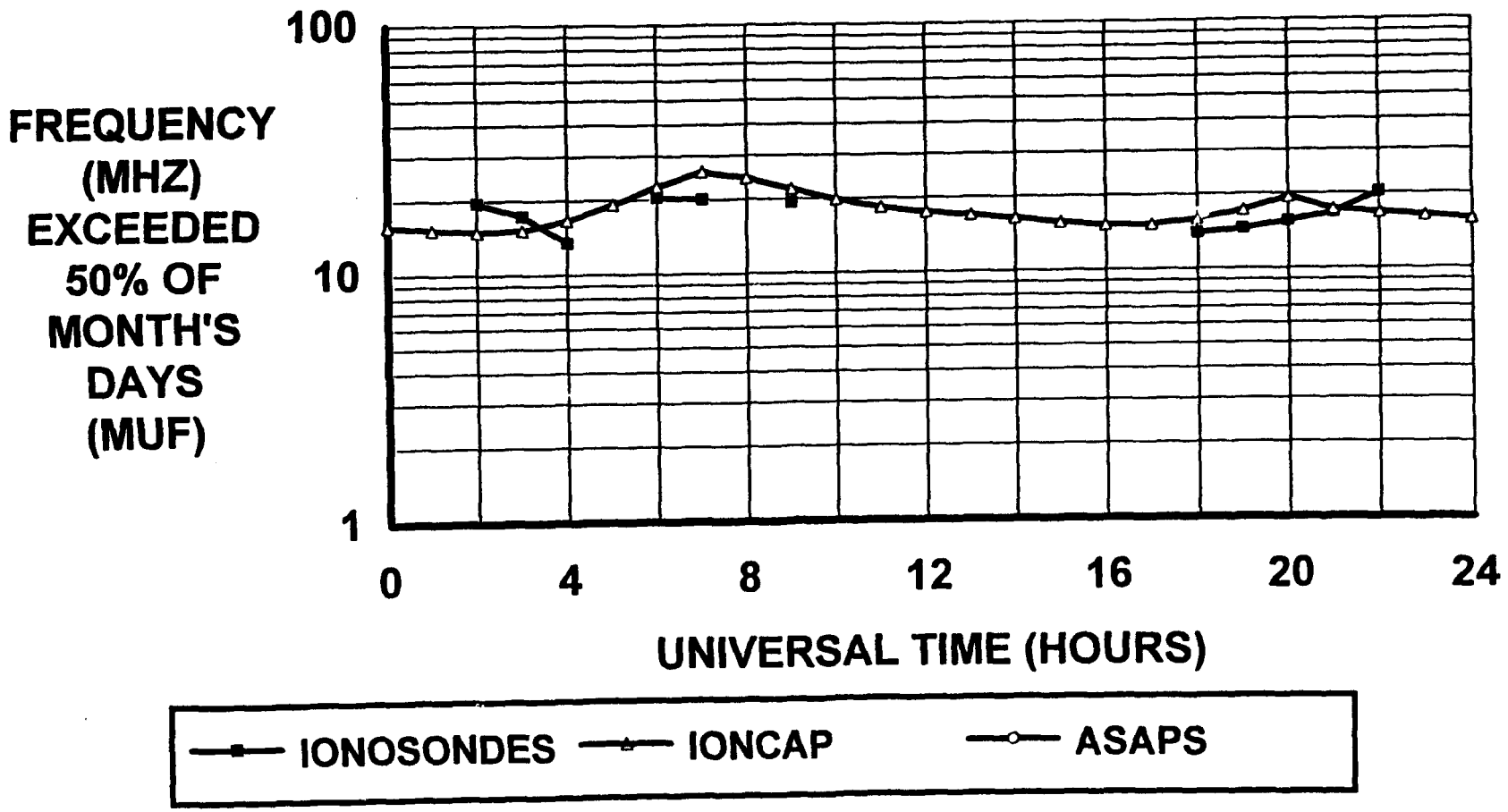
TABLE B.3 INDEX TO FREQUENCY AVAILABILITY PLOTS

		FOT		MUF		HPF	
LINK	RANGE	MONTH	PAGE	MONTH	PAGE	MONTH	PAGE
DIKSON TO RESOLUTE BAY	7100 KM	JAN	B-3	JAN	B-4	JAN	B-5
		FEB	B-6	FEB	B-7	FEB	B-8
		MAR	B-9	MAR	B-10	MAR	B-11
		APR	B-12	APR	B-13	APR	B-14
		MAY	B-15	MAY	B-16	MAY	B-17
		JUN	B-18	JUN	B-19	JUN	B-20
CAPE SCHMIDT TO [illegible]	9100 KM	MAR	B-22	MAR	B-23	MAR	B-24
		JUL	B-25	JUL	B-26	JUL	B-27
		SEP	B-28	SEP	B-29	SEP	B-30
		DEC	B-31	DEC	B-32	DEC	B-33

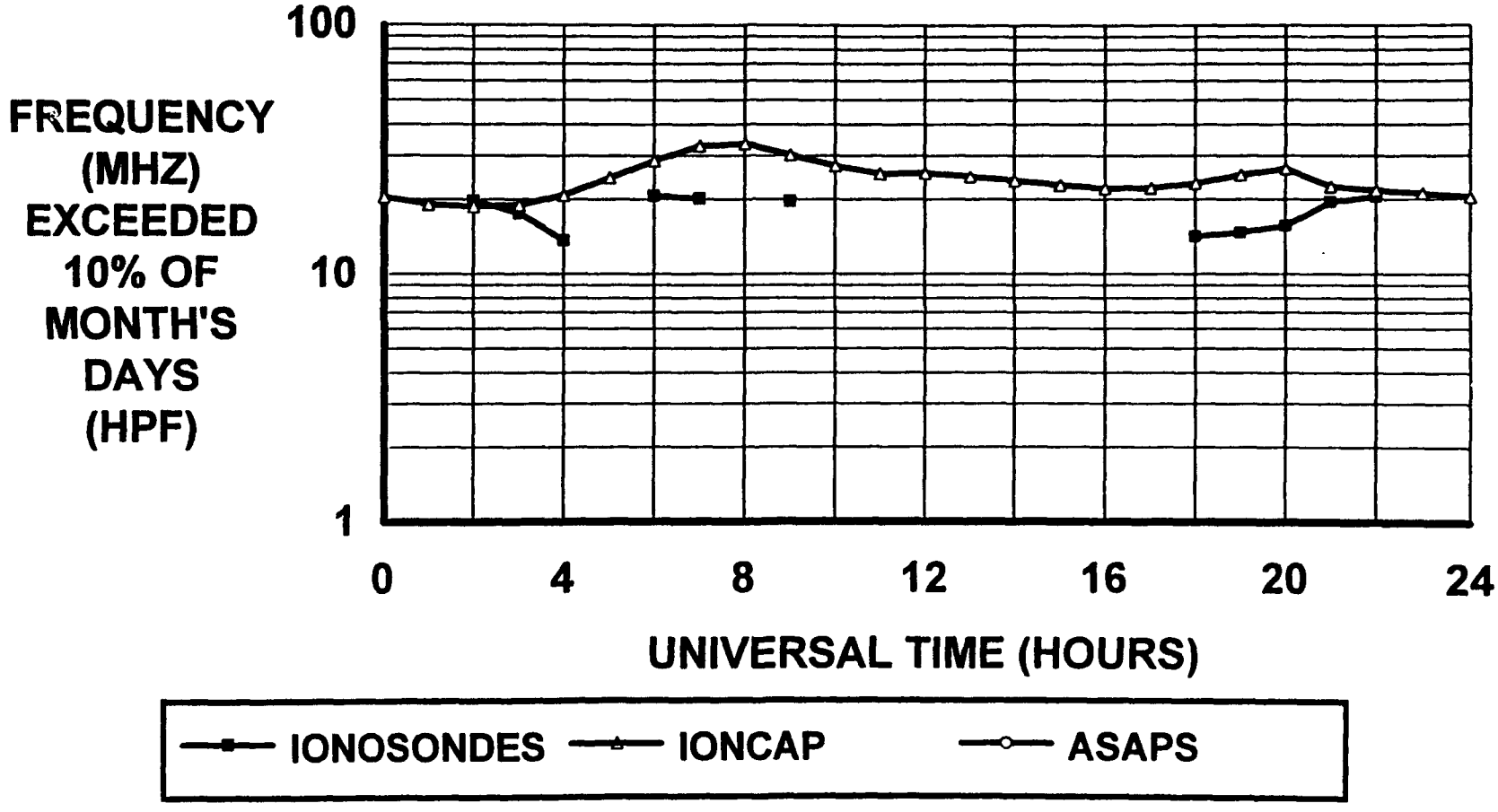
FOT COMPARISON MAR 1991 SSN: 181 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA

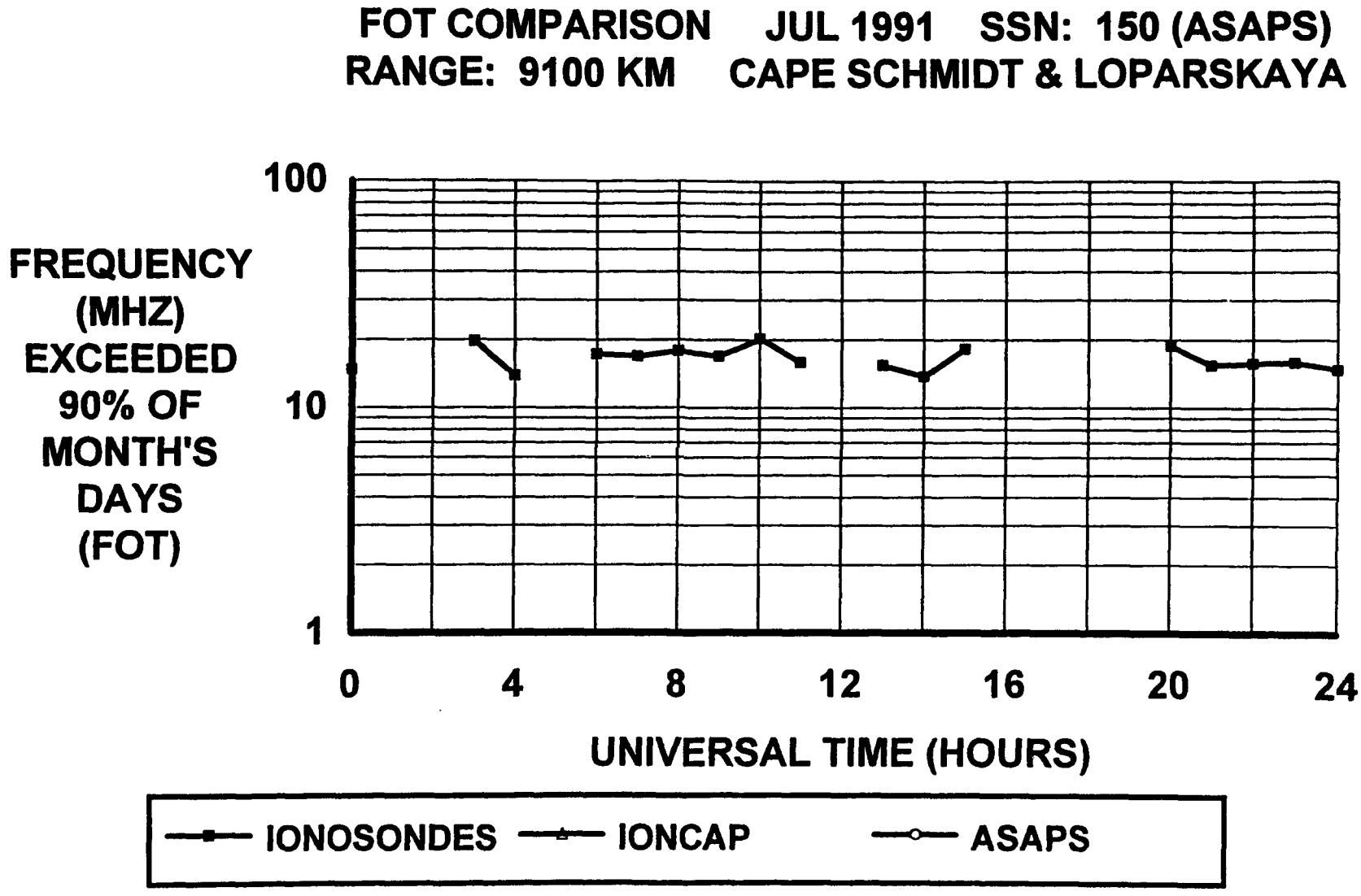


MUF COMPARISON MAR 1991 SSN: 181 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA

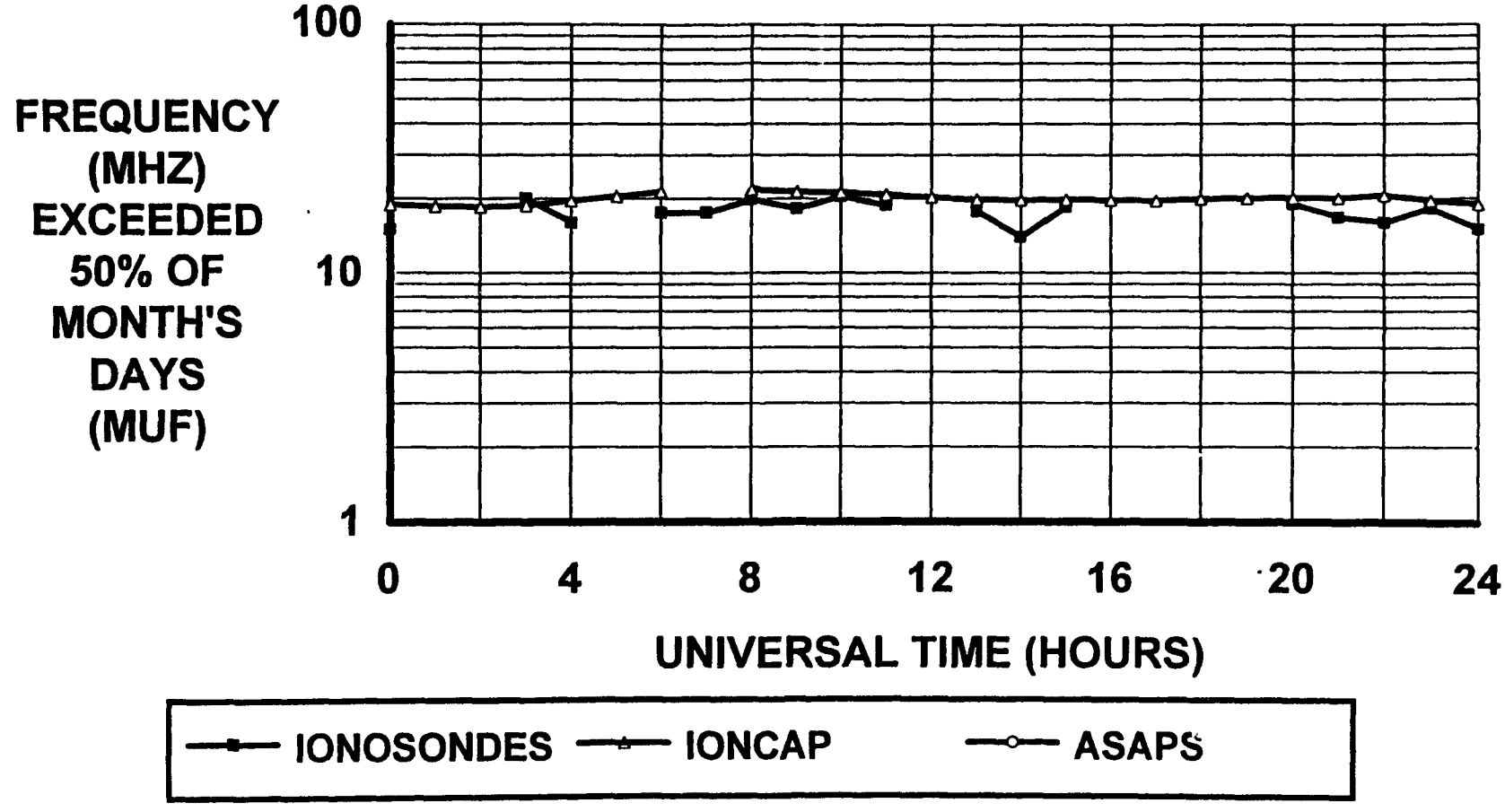


HPF COMPARISON MAR 1991 SSN: 181 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA

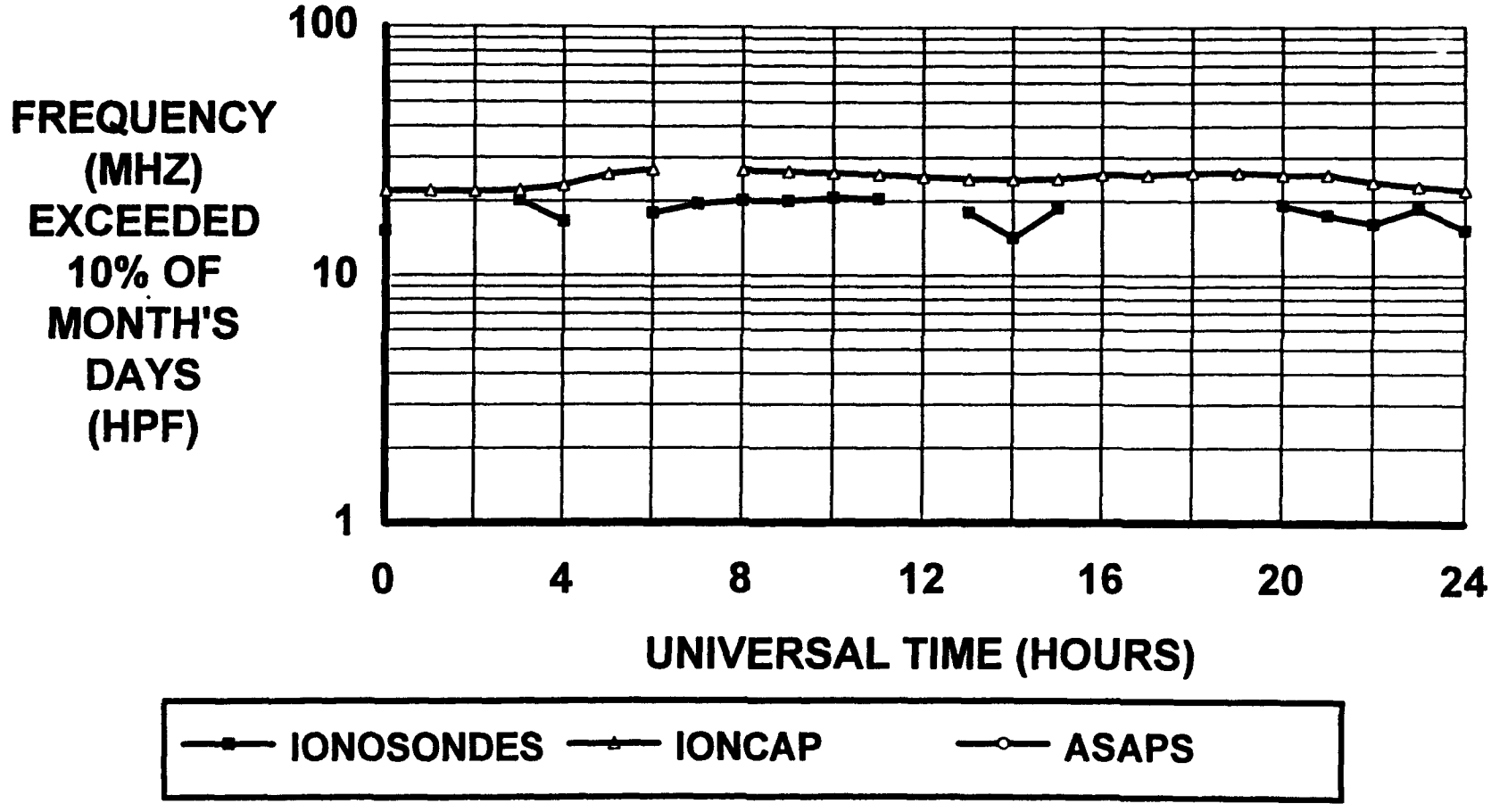



FOT COMPARISON JUL 1991 SSN: 150 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA
FREQUENCY (MHZ) EXCEEDED 90% OF MONTH'S DAYS (FOT)
100
10
1
0
4
8
12
16
20
24
UNIVERSAL TIME (HOURS)
IONOSONDES
IONCAP
ASAPS

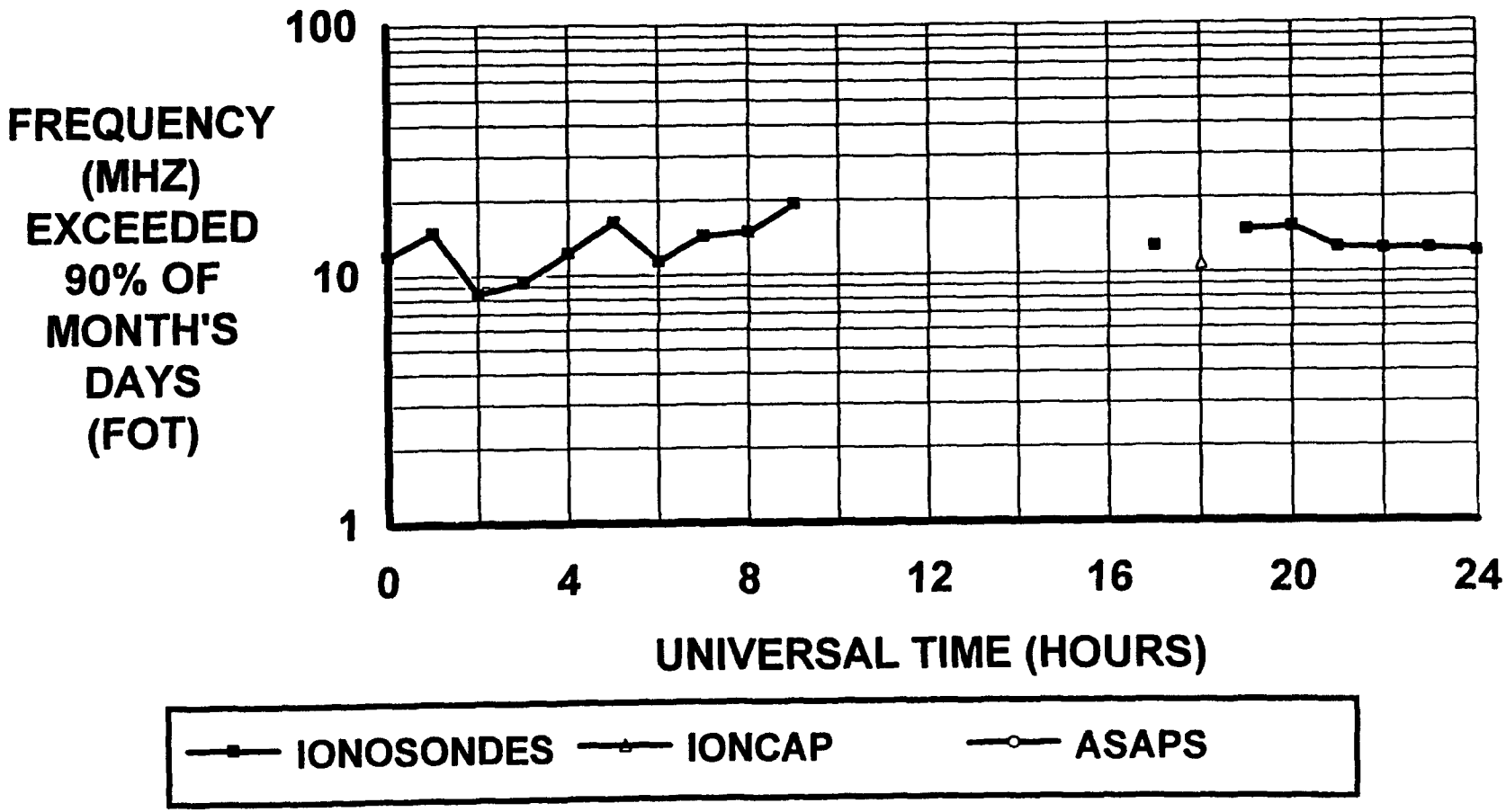
MUF COMPARISON JUL 1991 SSN: 150 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA



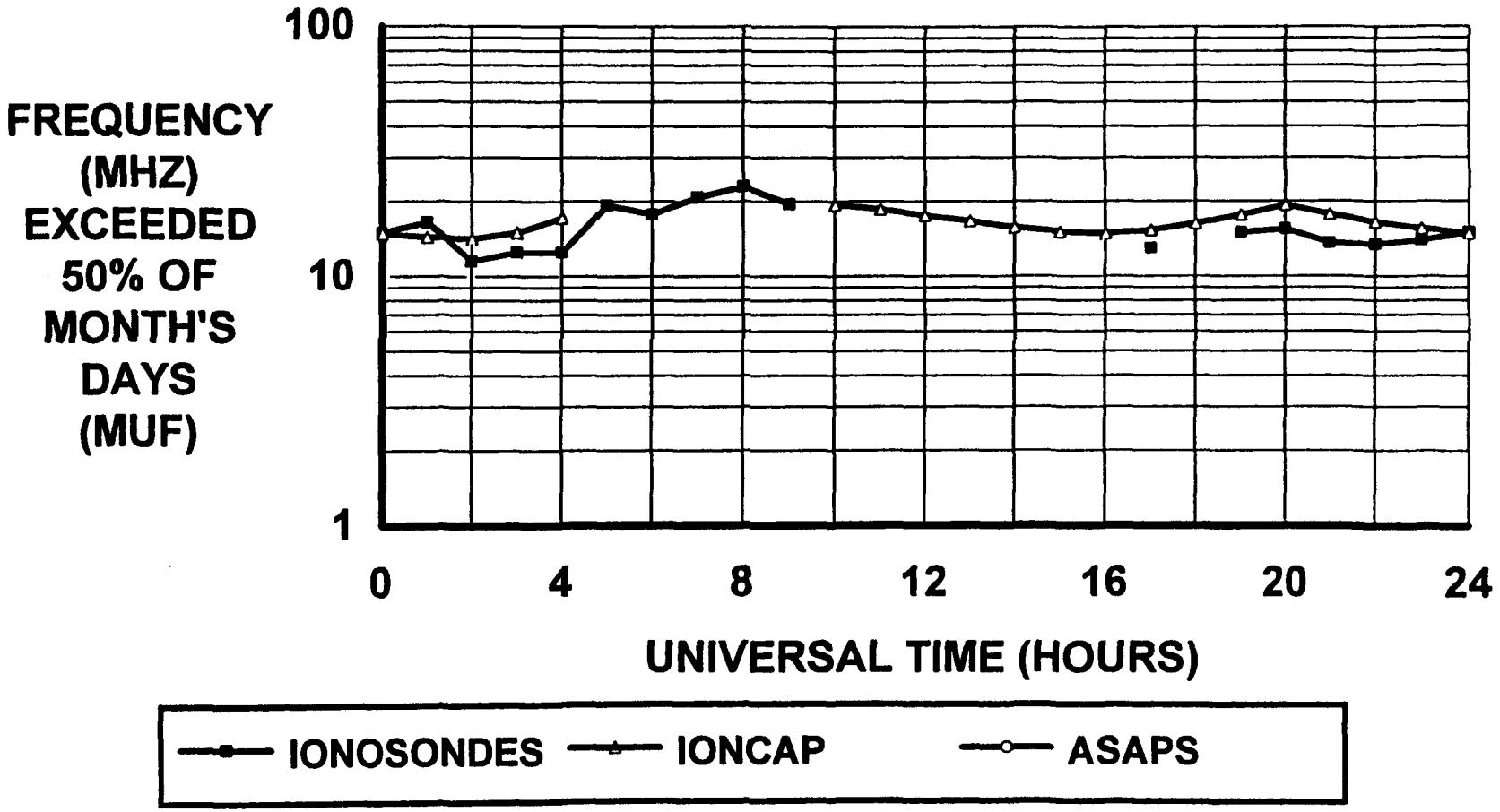
HPF COMPARISON JUL 1991 SSN: 150 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA



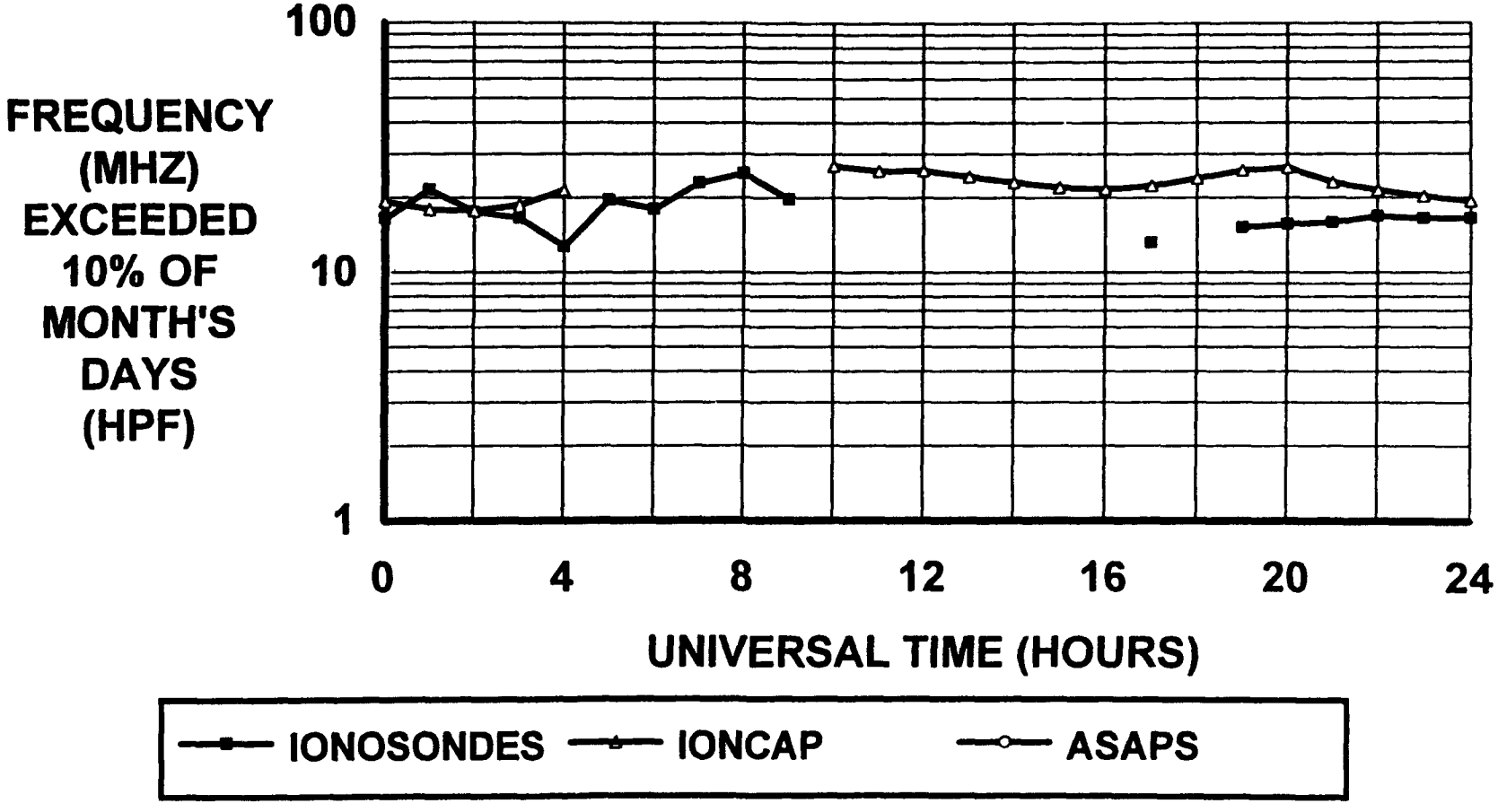
FOT COMPARISON SEP 1991 SSN: 148 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA



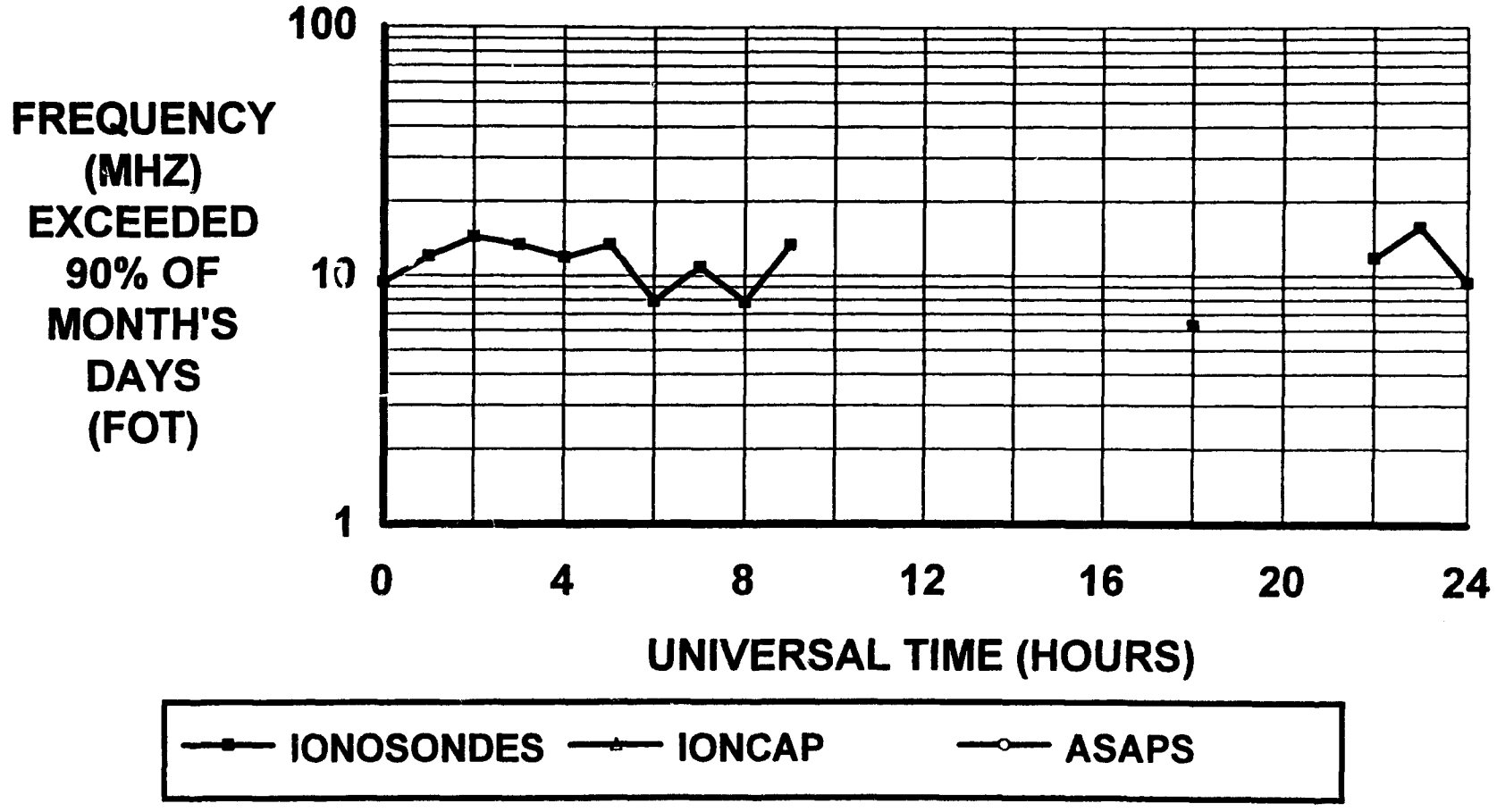
MUF COMPARISON SEP 1991 SSN: 148 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA



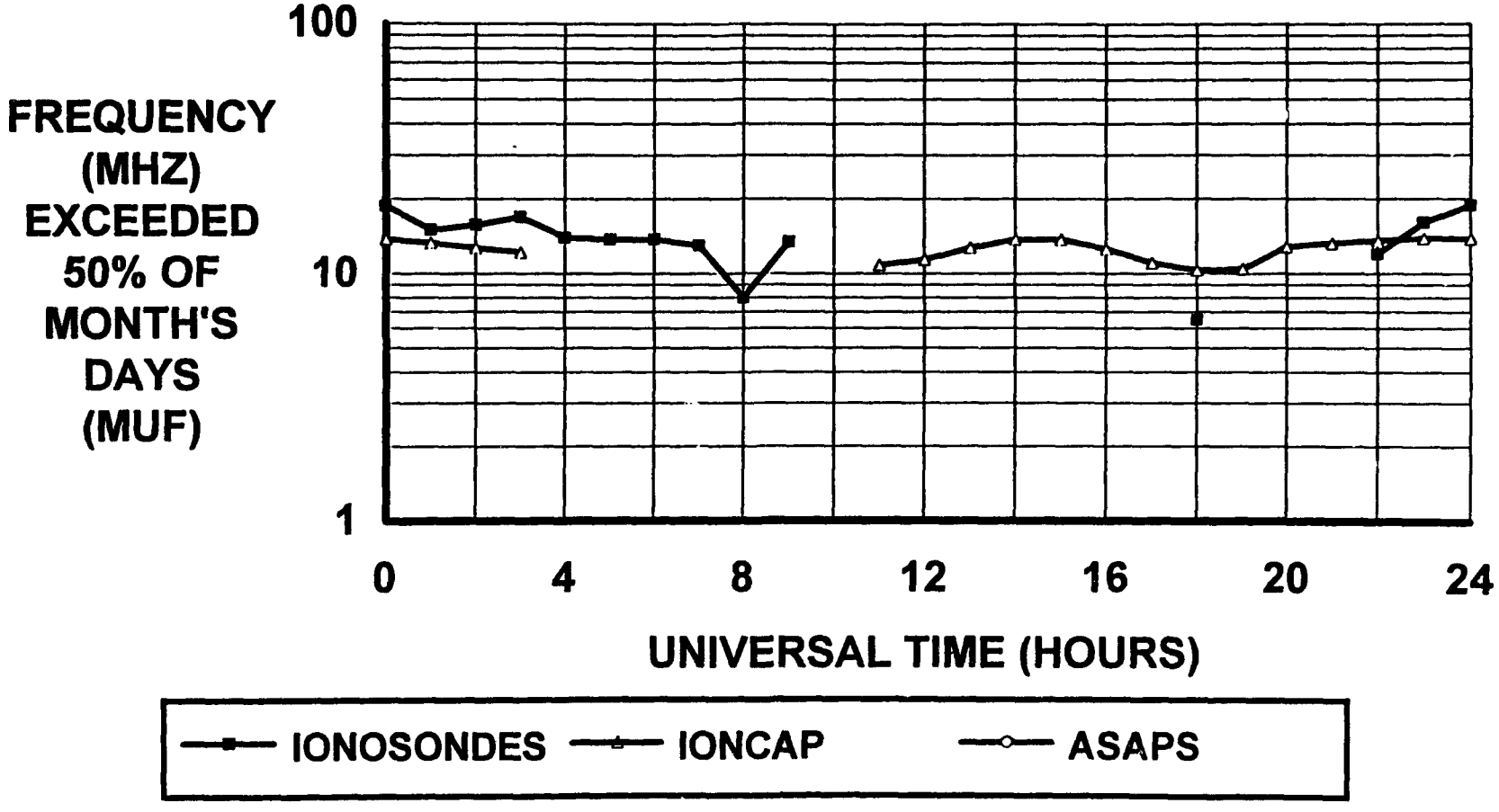
HPF COMPARISON SEP 1991 SSN: 148 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA



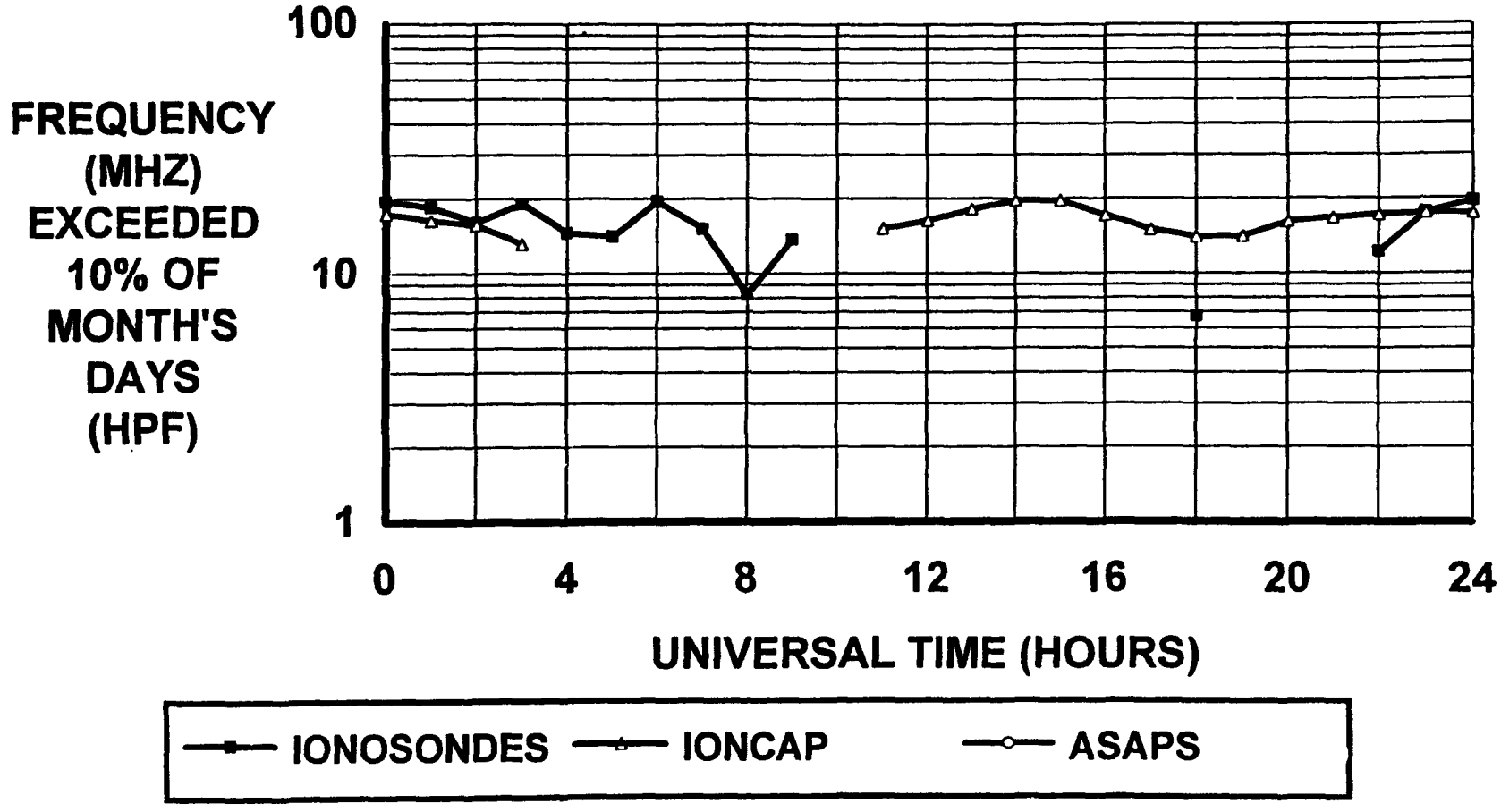
FOT COMPARISON DEC 1991 SSN: 134 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA



MUF COMPARISON DEC 1991 SSN: 134 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA



HPF COMPARISON DEC 1991 SSN: 134 (ASAPS)
RANGE: 9100 KM CAPE SCHMIDT & LOPARSKAYA



INITIAL DISTRIBUTION LIST

Addressee	No. of Copies
Defense Technical Information Center	12
Australian Antarctic Division (P. Yates, P. Magill)	2
ENEA (Dr. R. Cervellati)	1
IFRTP (Dr. C. Gillet)	1
Rome Laboratory (D. Spector)	1
Communications Research Centre (L. M. Boucher)	1
Defence Research Agency (Dr. P. W. Braddock)	1
U.S. Coast Guard Research and Development Center (CDR. J. John)	1
Science Applications International Corporation (R. Desourdis)	1
IPS Radio and Prediction Services (Dr. P. Wilkinson)	1
Defence Science & Technology Organisation (J. Tilbrook, Dr. S. C. Cook, Dr. R. Clarke)	3
National Science Foundation (P. Smith)	1
Chief of Naval Operations (CDR Ruud, LT Green)	2
National Geophysical Data Center (R. Conkright)	1
Capstone Systems (Dr. L. McNamara)	1