



A case-study analysis of surface/upper-air coupling in southeastern Montana
by Orville Dennis Green

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in
Earth Science

Montana State University

© Copyright by Orville Dennis Green (1986)

Abstract:

Rawinsonde data collected during the Cooperative Convective Precipitation Experiment (CCOPE) were analyzed to search for evidence of surface/upper-air coupling in southeastern Montana. CCOPE, an outgrowth of the HIPLEX-I weather modification project, was conducted near the semi-arid region of Miles City, Montana, during the summer of 1981. Five raw insonde stations were used in a limited station analysis scheme to examine three periods possessing varying degrees of convection.

The case-studies covered a severe-storm period (June 12-13), which produced radar reflectivities in excess of 50 dBz and a relatively large amount of precipitation; an intermediate convective period (June 20-21) characterized by the occurrence of small, semi-isolated cumuli selected as experimental units for the HIPLEX-I weather modification experiment; and a convectively benign period (July 30-31) with little or no visible convection. Surface/upper-air coupling during these periods was investigated by comparing vertical plots of areally averaged primary and derived meteorological parameters with corresponding temporal surface patterns.

Results suggest that surface/upper-air coupling, manifest as downward-propagating sinusoidal convergence and vorticity patterns, is more clearly established during the severe-storm period, evident to a lesser degree during the intermediate convective period, and absent during the benign period. Analogous analyses using nearby surface stations indicate that these patterns are reflected on the surface. However, the strongest evidence of surface/upper-air coupling based on surface information alone appears to be surface convergence and relative vorticity couplets of like sign.

Other features of the severe-storm period include a low-level jet with significant moisture advection, and an unstable total specific atmospheric energy profile aloft. These features were less striking or absent on the other case-study days.

A CASE-STUDY ANALYSIS OF SURFACE/UPPER-AIR COUPLING
IN SOUTHEASTERN MONTANA

by
Orville Dennis Green

A thesis submitted in partial fulfillment
of the requirements for the degree

of
Master of Science
in
Earth Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

June 1986

APPROVAL

of a thesis submitted by

Orville Dennis Green

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

20 May 1986

Date

James A. Kernbach, Jr.

Chairperson, Graduate Committee

Approved for the Major Department

20 May 86

Date

Stephen G. A. R.

Head, Major Department

Approved for the College of Graduate Studies

June 11, 1986

Date

Henry L. Parsons

Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Director of Libraries when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Orville Green

Date May 19, 1986

ACKNOWLEDGEMENT

For the many hours of patient encouragement and direction, I express my grateful appreciation to Dr. James A. Heimbach, Jr. I am also grateful for the contributions of Dr. Steve Custer, Dr. Joe Ashley, and Dr. Kathy Hansen-Bristow, who provided several useful comments during their critical review of this thesis. Mrs. Ann Parker was of considerable help in the preparation of the final copy of this manuscript.

This research was funded by the Experimental Meteorology Branch, National Science Foundation, Grant ATM-8209836.

TABLE OF CONTENTS

	Page
1. INTRODUCTION	1
Upper-Air Measurements	1
The Cooperative Convective Precipitation Experiment.	2
Focus of Current Study	4
Thesis Statement	5
2. HISTORICAL BACKGROUND.	6
The Thunderstorm Project	6
Relevant Surface Research.	8
Relevant Upper-Air Research.	10
Relevant Research in Southeastern Montana.	11
3. DATABASE DESCRIPTION	12
4. PROCEDURES	16
Sequence of Operations	16
Data Adjustments	19
Interpolation.	19
Balloon Drift.	20
Surface/Upper-Air Comparisons.	20
Network Differences.	20
Coordinate Reference Frames.	21
Data Processing Methods.	22
Evaluation of Meteorological Parameters.	22
Vapor Pressure	22
Absolute Humidity.	23
Specific Humidity.	23
Mixing Ratio	23
Relative Humidity.	23
Virtual Temperature.	24
Potential Temperature.	24
Equivalent Temperature	24
Equivalent Potential Temperature	24
Total Specific Atmospheric Energy.	25
Horizontal Kinematic Convergence	25
Vertical Component of the Relative Vorticity Vector.	25
Horizontal Moisture Convergence.	26
Vertical Velocity.	26

TABLE OF CONTENTS--Continued

	Page
5. CASE-STUDY ANALYSES.	27
Case-Study : The Severe-storm Period (12-13 June 1981)	29
General Meteorological Conditions During Severe-storm Case	29
Surface Patterns During the Severe-storm Case.	32
Surface Indications Prior to Precipitation Event #1.	34
Surface Indications During Precipitation Event #1.	34
Surface Indications During Precipitation Event #2.	38
Surface Indications During Precipitation Event #3.	38
Summary of Major Surface Features During Severe-storm Case	39
Upper-Air Profiles During the Severe-storm Case.	39
Surface/Upper-Air Coupling During the Severe-storm Case.	52
Case-Study : The HIPLEX-I Period (20-21 June 1981)	58
General Meteorological Conditions During the HIPLEX-I Case	58
Surface Patterns During the HIPLEX-I Case.	58
Upper-Air Profiles During the HIPLEX-I Case.	61
Surface/Upper-Air Coupling During the HIPLEX-I Case.	67
Case-Study : The Benign Period (30-31 July 1981)	71
General Meteorological Conditions During the Benign Case	71
Surface Patterns During the Benign Case.	71
Upper-Air Profiles During the Benign Case.	75
Surface/Upper-Air Coupling During the Benign Case.	75
6. SUMMARY AND CONCLUSIONS.	81
REFERENCES CITED	85
APPENDICES.	90
APPENDIX A - RAWINSONDE PROGRAM LISTINGS	91
Rawinsonde mainline programs	92
RAWIN task building executive.	92
RAWIN job processing executive	94
RAWIN file interpolation module.	97
RAWIN plot module.	100
RAWIN Plot Module with Clip options.	103
Rawinsonde Subprogram Units.	108
Bubble sort.	108
File deletion.	109
Drive number	110
U,V wind components.	110
Hypsometric evaluation	111
Equivalent potential temperature	111
Equivalent temperature	112
Potential temperature.	112
Virtual temperature.	113
Relative humidity.	113

TABLE OF CONTENTS--Continued

	Page
Mixing Ratio	114
Specific humidity.	114
Vapor pressure	115
Encode an integer *1 array	116
Average RAWIN station values	117
RAWIN sounding file dump	117
Set RAWIN processing flags	119
Fetch sounding data for all required stations.	121
Get fetch list	122
Fetch sounding data.	128
Access index file.	130
Fetch module common initialization	132
RAWIN jobfile access	136
RAWIN job processing initialization.	137
Calculate MSL heights.	138
Table interpolation.	139
Input operation code	142
Set plot specification	143
Calculate abscissa parameter	144
Input abscissa parameter	149
Calculate ordinate value	150
Input ordinate parameter	152
Input station combination code	154
Encode station ids	155
Input station combinations	156
Output RAWIN table	157
Temporary communication files.	164
Set default task specifications.	165
Set task specification	166
Output task specifications	167
Read task specifications	170
Write task specifications.	172
RAWIN task building initialization	173
Directory search	181
Convert wind components into direction and speed	182
Real linear interpolation.	183
Miscellaneous FORTRAN routines	184
Miscellaneous MACRO ASSEMBLER routines	191
Rawinsonde INCLUDE Files	197
 APPENDIX B - LSA PROGRAM LISTINGS.	 203
LSA Front-end.	204
PMKIN mainline	204
Option titles.	212
Segmentation communication dummy	213
Date/time and time increment	213
Kinematic convergence.	214
Area determination	215

TABLE OF CONTENTS--Continued

	Page
Pressure change.	216
Lambert conformal projection distances	217
Vorticity.	218
Convert wind components into direction and speed	219
Average of N parameters.	219
Specific humidity and mixing ratio	220
Potential and equivalent potential temperature	220
Sea level pressure	221
Moisture convergence	221
Running means.	222
Total specific atmospheric energy.	223
LSA Intermediate Segment	224
PMKIN plot preprocessor.	224
Maximum/minimum values	229
LSA Back-end	231
PMKIN plot module.	231
APPENDIX C - PLOT PROGRAM LISTINGS	232
Plot Hardcopies.	233
Three plots per page	239
One plot per page.	252
Miscellaneous Plotting Routines.	261

LIST OF TABLES

Table	Page
1. The location and elevation of the CCOPE rawinsonde sites (BUREC, . .	14
2. Divergence calculations with and without balloon drift at the 150 mb.	20
3. Point comparisons between upper-air and surface kinematic parameters.	56

LIST OF FIGURES

Fig.	Page
1. CCOPE surface mesonet, radar, and rawinsonde sites	13
2. Procedures for construction of the rawinsonde database on the IBM PC .	18
3. Kinematic convergence using 4 stations (dashed line) and 3 stations. .	28
4. Relative vorticity for the southwest quadrant (solid line) and the. .	30
5. Surface and 500 mb synoptic maps at 1200 GMT on 12 June 1981, with 24.	31
6. Precipitation, specific humidity, and relative humidity for the. . . .	33
7. Ambient temperature, station pressure, and total specific atmospheric.	35
8. Wind speed and wind direction for the southwest surface quadrant . . .	36
9. Kinematic convergence, moisture convergence, and relative vorticity. .	37
10. Specific humidity at 2000 GMT (solid line) and 2340 GMT (dashed. . . .	41
11. Relative humidity at 2000 GMT (solid line) and 2340 GMT (dashed. . . .	42
12. Ambient temperature at 2000 GMT (solid line) and 2340 GMT (dashed. . .	43
13. Equivalent potential temperature at 2000 GMT (solid line) and 2340 . .	44
14. Total specific atmospheric energy at 2000 GMT (solid line) and 2340. .	45
15. Wind speed at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 .	47
16. Wind direction at 2000 GMT (solid line) and 2340 GMT (dashed line) . .	48
17. Vertical velocity at 2000 GMT (solid line) and 2340 GMT (dashed. . . .	49
18. Kinematic convergence at 2000 GMT (solid line) and 2340 GMT (dashed. .	50
19. Moisture convergence at 2000 GMT (solid line) and 2340 GMT (dashed . .	51
20. Relative vorticity at 2000 GMT (solid line) and 2340 GMT (dashed . . .	53
21. Kinematic convergence (solid line) and relative vorticity (dashed. . .	54

LIST OF FIGURES—Continued

22. Kinematic convergence (solid line) and relative vorticity (dashed. . .	55
23. Surface and 500 mb synoptic maps at 1200 GMT on 20 June 1981, with . .	59
24. Specific humidity and relative humidity for the southwest surface. . .	60
25. Ambient temperature and total specific atmospheric energy for the. . .	62
26. Kinematic convergence, moisture convergence, and relative vorticity. .	63
27. Specific humidity at 2130 GMT (solid line) and 2344 GMT (dashed. . . .	64
28. Total specific atmospheric energy at 2130 GMT (solid line) and 2344. .	65
29. Vertical velocity at 2130 GMT (solid line) and 2344 GMT (dashed. . . .	66
30. Kinematic convergence at 2130 GMT (solid line) and 2344 GMT (dashed. .	68
31. Moisture convergence at 2130 GMT (solid line) and 2344 GMT (dashed . .	69
32. Relative vorticity at 2130 GMT (solid line) and 2344 GMT (dashed . . .	70
33. Surface and 500 mb synoptic maps at 1200 GMT on 30 July 1981, with . .	72
34. Specific humidity, relative humidity, and total specific atmospheric .	73
35. Kinematic convergence, moisture convergence, and relative vorticity. .	74
36. Specific humidity at 1830 GMT (solid line), 2340 GMT (long dashed. . .	76
37. Total specific atmospheric energy at 1830 GMT (solid line), 2340 GMT .	77
38. Kinematic convergence at 1830 GMT (solid line), 2340 GMT (long	78
39. Moisture convergence at 1830 GMT (solid line), 2340 GMT (long dashed .	79
40. Relative vorticity at 1830 GMT (solid line), 2340 GMT (long dashed . .	80

ABSTRACT

Rawinsonde data collected during the Cooperative Convective Precipitation Experiment (CCOPE) were analyzed to search for evidence of surface/upper-air coupling in southeastern Montana. CCOPE, an outgrowth of the HIPLEX-I weather modification project, was conducted near the semi-arid region of Miles City, Montana, during the summer of 1981. Five rawinsonde stations were used in a limited station analysis scheme to examine three periods possessing varying degrees of convection.

The case-studies covered a severe-storm period (June 12-13), which produced radar reflectivities in excess of 50 dBz and a relatively large amount of precipitation; an intermediate convective period (June 20-21) characterized by the occurrence of small, semi-isolated cumuli selected as experimental units for the HIPLEX-I weather modification experiment; and a convectively benign period (July 30-31) with little or no visible convection. Surface/upper-air coupling during these periods was investigated by comparing vertical plots of areally averaged primary and derived meteorological parameters with corresponding temporal surface patterns.

Results suggest that surface/upper-air coupling, manifest as downward-propagating sinusoidal convergence and vorticity patterns, is more clearly established during the severe-storm period, evident to a lesser degree during the intermediate convective period, and absent during the benign period. Analogous analyses using nearby surface stations indicate that these patterns are reflected on the surface. However, the strongest evidence of surface/upper-air coupling based on surface information alone appears to be surface convergence and relative vorticity couplets of like sign.

Other features of the severe-storm period include a low-level jet with significant moisture advection, and an unstable total specific atmospheric energy profile aloft. These features were less striking or absent on the other case-study days.

CHAPTER 1

INTRODUCTION

Upper-Air Measurements

The atmosphere aloft is routinely sampled by instrumented balloons called rawinsondes. These balloons are released at the surface and transmit the pressure, temperature, and humidity of the air through which the instrument is ascending. In addition, the wind speed and wind direction of the upper atmosphere are derived from balloon positions obtained by tracking the balloon from its release point to its position at the time the balloon bursts. When synoptic data from three or more proximate soundings are combined, estimates of other meteorological parameters are possible; e.g., the horizontal velocity divergence, the horizontal moisture convergence, the vertical component of the vorticity, and the vertical component of the wind.

The National Weather Service (NWS) schedules two soundings per day at 0000 and 1200 GMT (Greenwich Mean Time) from approximately 80 sites in the contiguous United States. The NWS sounding network is sufficient for identifying the migratory high and low pressure systems prevalent over the continental United States. However, it is inadequate for forecasting gross processes associated with even the larger convective events, such as those associated with squall lines (House, 1960), which have a characteristic length on the order of 10^4 - 10^5 m (Holton, 1972).

The ability of the synoptic-scale soundings to measure smaller convective events, such as an isolated thunderstorm having a characteristic length of from

1-50 km (Matthews, 1983), is even more limited. Yet the study of such mesoscale phenomena has been selected as a national objective for the atmospheric sciences by the National Academy of Sciences (1980); with a high priority on the identification of the mechanisms which trigger and sustain severe convective storms.

In order to investigate the upper-air environment affected by mesoscale convective complexes, a rawinsonde network that samples at the length and time scales of the event itself is required. Using Orlanski's (1975) scheme for classifying subsynoptic scale phenomena :

meso α : 200 - 2000 km, 6 - 36 h

meso β : 20 - 200 km, 1 - 6 h

meso γ : 2.0 - 20 km, 0.1 - 1 h

a rawinsonde network that would adequately measure isolated convective phenomena would require a network spacing in the meso γ - meso β scale.

The Cooperative Convective Precipitation Experiment

Just such a rawinsonde network was established within a surface mesoscale network (mesonet) for the Cooperative Convective Precipitation Experiment (CCOPE), conducted in the semi-arid region east of Miles City, Montana during the summer of 1981. CCOPE was a cooperative venture funded primarily by the U.S. Bureau of Reclamation (BUREC) and the Convective Storms Division (CSD) of the National Center for Atmospheric Research (NCAR). Other federal agencies, universities, and private organizations, as well as personnel from five different countries participated (BUREC, 1982). CCOPE was designed to evaluate "the potential of convective cloud modification techniques and [to predict] convective precipitation and associated severe weather" (Dennis, et al.,

1984). The CCOPE project was initiated because the resources available to the High Plains Experiment (HIPLEX) were insufficient to adequately resolve questions about the physical processes attendant with convective events.

HIPLEX, which itself was essentially a continuation of the BUREC's Project Skywater experiment, was implemented "to reduce the scientific uncertainties concerning both natural and artificially modified precipitation processes in warm season convective clouds of the High Plains region of the United States" (Smith, et al., 1984). The experimental design of HIPLEX embodied three phases: an exploratory study, experiments on single clouds and cloud systems, and an area experiment.

The exploratory phase of HIPLEX was conducted in Montana, Kansas, and Texas in the mid-1970s. By early 1979, the HIPLEX scientists had obtained enough information to develop a specific experimental plan for Phase 2; i.e., the randomized seeding of small, semi-isolated, cumulus congestus clouds. HIPLEX-I, the initial part of Phase 2, was conducted during the summers of 1979 and 1980. The number of test cases occurring during these two summers was less than had been climatologically expected and, rather than extend HIPLEX-I, the decision was made to conduct CCOPE.

The resulting cooperative effort established the largest mesonet in the world: 123 surface stations covering 22400 square kilometers. In addition, 14 aircraft, up to 8 rawinsonde units, 1 BUREC Skywater radar, and a network of 7 other radars having Doppler capability were operated during CCOPE (BUREC, 1982). The information derived from this massive data gathering network would have contributed to subsequent phases of the HIPLEX project; however, funding for HIPLEX was terminated in 1981.

Focus of Current Study

The cessation of HIPLEX before analysis of the CCOPE data was completed left many of the fundamental questions about convective processes unanswered. Rather than relegate the CCOPE database to the archives, however, funds were solicited from the National Science Foundation to examine triggering mechanisms and other meteorological features which would influence the development and sustenance of convective complexes (Heimbach and Engel, 1981). A convective complex was defined to be a "radar echo that exceeded 30 dBz and reached an altitude of 9 km MSL [Mean Sea Level] or higher during its lifetime, but not necessarily simultaneously" (Heimbach and Super, 1980).

One of the primary unresolved problems addressed in the proposal was the identification of successful predictor-covariates of convection. The number of potential candidates was impressive. For example, Acherman, et al. (1976) selected 47 promising candidates from a list of 199 in their summary of the design for Phase 2 of HIPLEX. Heimbach and Engel (1981), however, selected 18 covariate candidates for the statistical part of their study. These were organized into two groups: those found reasonably successful in previous studies, such as surface kinematic and moisture convergence, vorticity, and mixing ratio; and a set of newly proposed covariates including latent energy and convective destabilization.

The fundamental assumption underlying the statistical analysis is that the surface boundary layer is coupled to convective events aloft. The assumption of coupling provides the physical link that allows a causal explanation to be applied to the correlation. Therefore, the conditions under which this assumption is valid need to be defined.

Thesis Statement

The purpose of this study is to examine the CCOPE rawinsonde database for evidence of surface/upper-air coupling. Specifically, the hypothesis is advanced that surface mesoscale features commonly associated with convection are coupled to convective phenomena aloft in southeastern Montana. The conditions under which surface/upper-air coupling occurs is determined by investigating three periods possessing different degrees of convection. The case-studies consist of a severe-storm period (June 12-13) producing radar reflectivities in excess of 50 dBz with a large quantity of precipitation; a intermediate convective period (June 20-21) possessing the small, semi-isolated cumulus congestus clouds selected as experimental units for the HIPLEX-I project; and a convectively benign period (July 30-31) with little or no visible convection.

CHAPTER 2

HISTORICAL BACKGROUND

The Thunderstorm Project

The Thunderstorm Project, summarized in a notable and enduring study by Byers and Braham (1949), arose out of a need to better understand the extreme hazard that these convective events posed to aircraft, especially since pilots could no longer restrict their flights to good weather. The fortuitous close of the war allowed the conversion of wartime equipment and personnel to peacetime research, and enabled the study to be conducted on a more complete scale than at any prior time.

The Thunderstorm Project studied thunderstorms occurring mostly in maritime-tropical air masses near Orlando, Florida and Wilmington Ohio. A major feature identified during the study was "convection units having similar properties and characteristics" (Byers and Braham, 1949) localized within the thunderstorm. These regions were defined as "cells". The lifetime of thunderstorm cells could exceed 30 minutes, during which time they tended to retain their identity and move with the body of the storm.

Three stages characterized the life of a thunderstorm cell : the cumulus stage, dominated by an updraft throughout the cell; the mature stage, in which a downdraft is initiated in part of the cell by viscous drag from falling water droplets, and subsequently enhanced by the entrainment of the colder ambient air surrounding the cell; and the dissipating stage, in which the downdraft, growing at the expense of the updraft, eventually cuts the updraft off from its

source region. With the dissipation of the updraft and subsequent removal of the source of rainfall, the cell eventually dies.

Those surface features attending the life cycle of the thunderstorm cell were also described. The cumulus stage, which normally lasted from 10 to 15 minutes, was accompanied by a slightly lowered surface pressure, and horizontal velocity convergence up to 1/2 h prior to the first radar echo. This surface convergence accounts for the "calm before the storm" in cases where the inflow opposes the prevailing winds.

By the time the cell develops into the mature stage, the cold-core downdraft carrying the rain reaches the surface, and flows outward from the rainfall area. The wind speed in the downstream portion of the cold air dome usually far exceeds the wind speed in the upstream portion, becoming an asymmetrical "gust-front" which travels ahead of and in the direction of the storm itself. This gust-front or "micro-cold front" (Byers and Braham, 1949) is accompanied by strong horizontal velocity divergence, a sudden surface temperature drop, and a sharp pressure rise. The heaviest rainfall was found to occur within the area of the maximum horizontal velocity divergence, but somewhat earlier due to the greater velocity of the water droplets with respect to the downdraft. The mature stage usually lasts for 15 to 30 minutes.

During the dissipating stage the precipitation gradually ceases, the surface pressure recovers from the sharp pressure jump, and the horizontal velocity divergence diminishes. The dissipating stage lasts for approximately 30 minutes.

Not all thunderstorm cells follow this pattern. For example, vertical wind shear may cause the downdraft region to occur outside of the initial cell, and thus prolong the life of the updraft. In fact, the cell may dissipate without

establishing the downdraft evident during the normal dissipating stage. On the other hand, some cumuli fail to develop into thunderstorms. Byers and Braham (1949) surmised that return settling of the environment around large thunderstorm cells may inhibit development of nearby clouds. Return settling refers to large-scale downward air motion compensating the localized updraft regions in developing cumuli.

Kinematic analysis of balloon ascents into isolated clouds revealed an interesting pattern of alternating convergence/divergence regimes aloft during ten of the seventeen soundings taken. Each of these storms had developed a downdraft with rain and showed low-level divergence under a layer of convergence, with divergence again above about 6 km. The low level divergence was a result of the downdraft, while the convergence aloft resulted from horizontal inflow of the ambient air through the cloud boundary. In the remaining seven balloon ascents, the rain had not begun at the surface, and the profiles showed convergence at all levels.

Relevant Surface Research

Noting that Byer and Braham's (1949) observation of the occurrence of surface convergence 20-30 min before the appearance of the first radar echo had received little attention since initially reported, Ulanski and Garstang (1978) studied the role that surface conditions, primarily the horizontal wind speed and direction, played in the life cycle of convective storms. The premise was that a time-persistent cellular pattern existed in the surface divergence and vorticity fields, and that these fields were correlated to convective events. Using data from the surface mesonet established during the Florida Area Cumulus Experiment (FACE) in 1971 and 1973 (see Woodley and Sax, 1976), objectively

analyzed maps of surface divergence and vorticity were obtained for 5-min intervals.

A direct relationship between total storm rainfall and the areal extent of the surface convergence was found. The size of the surface area of convergence was observed to be a critical factor in controlling the total amount of rainfall produced by a given convective storm. This is because the size of the surface convergence area determines the amount of moisture supplied to the storm. For intense convective events, the larger portion of surface moisture is supplied during the storm's mature stage.

The size of the initial convergence cell was also related to the point maximum rainfall. The size of the convergence cell was parameterized by defining a "spatial index" as "the shortest distance (km) from the center of the convergence cell to the zero isopleth of divergence. . . computed as an average distance over the first 15 min of the life of a center of convergence" (Woodley and Sax, 1976). A linear correlation coefficient of 0.72 was found between the spatial index and point rainfall.

A direct relationship was discovered between the storm maximum point rainfall and the length of time that surface convergence existed before the rain. Because of the lead time, which exceeded 90 min in the larger storms studied, it was concluded that the surface convergence caused the convective storm, and was not a result of it. Though not specifically mentioned as such, surface/upper-air coupling is suggested by the observation that "in each case the shower developed in the vicinity of the initial convergence cell" (Woodley and Sax, 1976). It was also observed that the initial field of positive convergence and vorticity changed to negative values during the time of maximum rainfall.

Relevant Upper-Air Research

Prior to HIPLEX, weather modification experiments did not recognize the need for mesoscale rawinsonde observations (Matthews, 1983). A significant study of a mesoscale rawinsonde network was, however, accomplished by Fankhauser (1969). He applied objective analysis techniques to a 9-station network in Oklahoma to "appraise the sounding system's limitations and reliability for resolving the mesoscale circulations associated with convective processes." The primary deficiencies identified in the rawinsonde system were the inherent temporal and spatial limitations in the sampling network, and assumed measurement and analysis errors. These errors included non-simultaneous release times, balloon drift, varying ascent rates, decreasing wind measurement accuracy with decreasing elevation angles (increasing height), and error accumulation with height due to vertical integration of the continuity equation (see Chapter 5). The effects of these deficiencies were minimized by spatial and temporal smoothing of the data reduction and analyses, using both objective and subjective techniques. However, the smoothing techniques and basic observational limitations did not allow downward extension to scales on the order of singular thunderstorm processes. Nevertheless, it was found that good qualitative agreement existed between the kinematic divergence and vertical motion, and thunderstorm location and intensity.

During the summer of 1976, a 5-station rawinsonde network was operated at the Texas HIPLEX site. Scoggins, et al. (1978) reported that moisture convergence, vertical motion, and vertical moisture flux proved to be good covariates of convection, though nearly all of the surface and upper-air variables showed some response to convective activity.

Relevant Research in Southeastern Montana

The foregoing studies were all accomplished outside the Montana High Plains. Doneaud, et al. (1983) performed a case-study analysis on the 1980 HIPLEX surface mesonet data collected in southeastern Montana to investigate the surface kinematic fields attendant with convective storms. It was found that convergence cell development preceeded radar echoes and rain, in agreement with earlier studies. An interesting feature of the convective clouds in southeastern Montana was that they received a greater moisture inflow and, thus, produced less rain efficiency than clouds with similar rainfall duration and amounts in humid climates.

CHAPTER 3

DATABASE DESCRIPTION

The CCOPE surface mesonet consisted of 96 BUREC PROBE (Portable Remote Observation of the Environment) stations, operated by the Montana Department of Natural Resources and Conservation, and 27 NCAR PAM (Portable Automated Mesonet) stations. Seventy-one of the PROBE stations were placed in a 9 X 8 square grid, with a station-to-station spacing of approximately 20 km. The remaining PROBE and all the PAM stations were established in a dense, irregular inner network under Doppler cover, with a station separation of approximately 7 km (see Fig. 1).

The BUREC PROBE stations transmitted hourly blocks of 5-min arithmetically averaged temperature, pressure, relative humidity, east and north wind components, and cumulative precipitation to the GOES-West (Geostationary Observational Earth Satellite). The information was then retransmitted to the Denver BUREC downlink, and to a backup site at NOAA-NESS in Wallops Island, Virginia (BUREC, 1982).

The NCAR PAM stations telemetered 1-min average dry-bulb temperature, wet-bulb temperature, pressure, east and north wind components, peak wind velocity during each minute, and the rainfall to the PAM base at the Miles City airport.

The supporting rawinsonde network consisted of 5 primary stations (see Table 1) operated by Texas A&M University (TAMU), a redundant system at Miles City operated by Western Weather Consultants, Inc. (WWCI), and 2 stations operated in late June and early July by the Alberta Research Council.

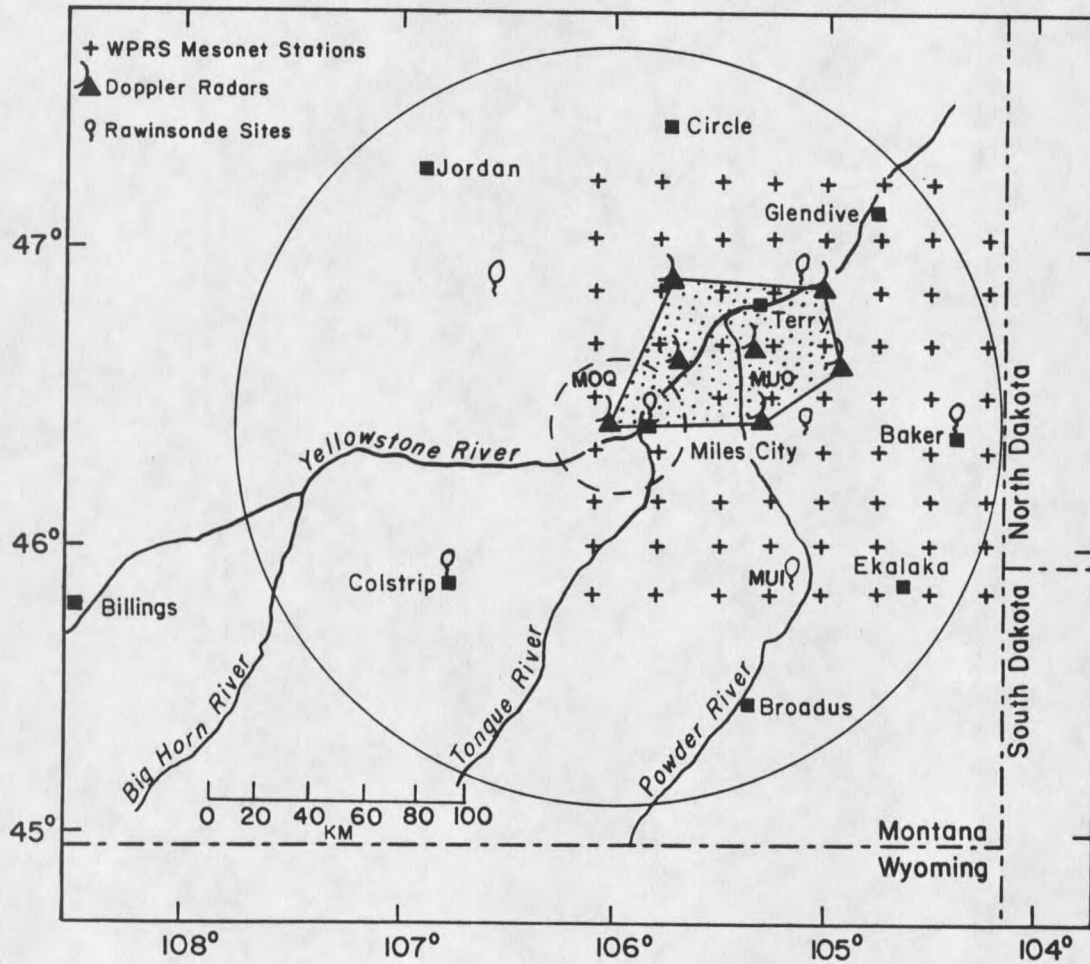


Figure 1. CCOPE surface mesonet, radar, and rawinsonde sites.

Table 1. The location and elevation of the CCOPE rawinsonde sites (BUREC, 1982)

Station	Elevation in meters	Latitude (N)			Longitude (W)		
		Deg	Min	Sec	Deg	Min	Sec
Baker	903	46	21	10.6	104	13	26.3
Glendive	803	47	05	00.0	105	06	42.6
Knowlton	954	46	20	13.8	105	04	40.0
Miles City	800	46	26	05.5	105	52	30.0
Powderville	866	45	45	20.3	105	06	07.0

TAMU took soundings on 54 days during 21 May to 6 August; usually at a frequency of one sounding every 1.5 h between 1800-0300 GMT (1200-2100 MDT [Mountain Daylight Time]), while WWCI took 2 soundings daily, 6 days a week on a non-rigid schedule consisting of one early in the morning (1145 GMT) and one in the early afternoon at 1930 GMT. The sounding data consisted of the temperature and relative humidity ordinates, the pressure, time after release for each pressure contact, and the azimuth and elevation angles at 30-second intervals. Normally, soundings were terminated at 100 mb, which is approximately 16 km.

The rawinsonde data selected for this study were collected by NCAR, who also performed the necessary data integrity tests and post-screening procedures. Wind information contained in this dataset had been lightly smoothed with a nine-point filter¹ and were interpolated to pressure-contact times where necessary. No additional screening or editing of these data were accomplished, other than simple spot-checks on the database using a hand calculator and the raw data contained on microfiche.

¹Weights were 0.3200156553, 0.2643210644, 0.1343767864, 0.0114831529, -0.0458900942, -0.0377520979, -0.0064753717, 0.0111929411, 0.0087357913

Radar information was used to assist in the identification of the case-study periods selected for this study. These data came from the BUREC 5.4 cm Skywater radar which was located about 3 km west of Miles City. The radar beam width was 1 deg, and scans were made from elevation angles 1 - 12 deg. Under normal operation, the radar scans were taken every 5 min; however, a rapid-scan mode was also available, which scanned every 3 min. These data were processed by the BUREC staff at Miles City (Heimbach and Engel, 1986).

CHAPTER 4

PROCEDURES

Sequence of Operations

The underlying premise for this study was that surface/upper-air coupling could be identified if vertical profiles of those parameters commonly associated with convective activity were generated. The features manifest in the vertical profiles would be evident at the surface as well, if coupling had been established. Thus, though the study would focus on the rawinsonde data, the surface database would have to be accessed as well.

The initial step was to inspect the CCOPE surface data for obvious errors and correct them. This was accomplished by referring to field notes and visually inspecting each day's data, which were available in graphical form. The types of errors observed during the visual inspection included transmission noise and obvious errors in the parameter traces. Two researchers independently reviewed each day's data to minimize oversights and personal bias. The necessary corrections were then entered into a recalibration file, which could be accessed when the surface data were processed. The recalibration file contained other corrections to the surface database which were not readily evident on the parameter traces, such as pressure recalibration factors from NCAR.

It was also presumed that the vertical profiles would vary according to the degree of convection. For example, Ulanski and Garstang (1978) observed that divergence values discriminated between convectively active and inactive

periods. Accordingly, it would be expected that the kinematic values aloft would show similar discrimination. Therefore, the vertical profiles would have to cover periods possessing different degrees of convection in order to determine the extent that surface/upper-air coupling depended on the magnitude of the convection. To this end, operational field notes and radar data were examined to select three case-study periods with high, intermediate, and low degrees of convection.

June 12-13 was selected to represent the period with high convective activity; i.e., the "severe-storm" period. During this period, a large convective complex traversed the mesonet, producing large amounts of precipitation with probable hail. In addition, a tornado watch had been issued for the area south of Miles City. For the intermediate, or "HIPLEX-I" period, June 20-21 was selected because it contained the isolated cumulus congestus clouds sought as experimental units during the HIPLEX-I weather modification experiment. A cloud-free period, July 30-31, was selected for the low, or "benign" period. The terms "severe-storm", "HIPLEX-I", and "benign" will be used in the remainder of this paper to reference the case-study periods.

Next, the rawinsonde data were obtained on magnetic tape from the BUREC, who had the responsibility of archiving all the CCOPE data. Procedures for downloading and accessing the rawinsonde data on an IBM PC were developed (see Fig. 2). Several man-weeks were spent developing algorithms that would both analyze the raw rawinsonde data and interface with existing (surface) software. After the BUREC datatape arrived, the rawinsonde program's processing schemes were tested against NCAR microfiche data. A discrepancy was discovered between the BUREC processed data and the NCAR microfiche data, even though the raw data values for the two datasets were similar.

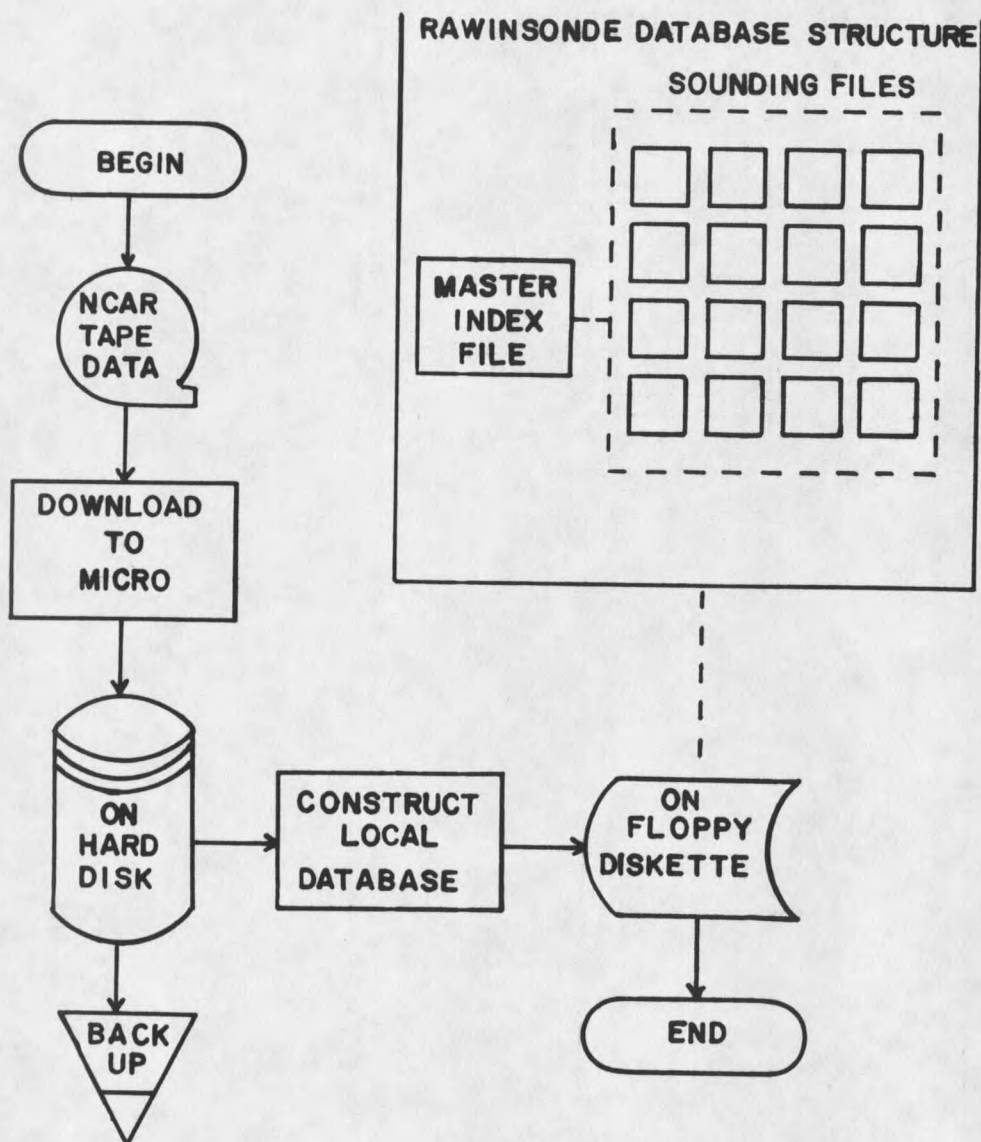


Figure 2. Procedures for construction of the rawinsonde database on the IBM PC.

Following procedures outlined by Retallack (1971), several hand calculations were performed on both datasets. It was discovered that an error existed in the BUREC data, which was eventually traced to an initialization error in BUREC's RAOB program. Rather than wait for BUREC to correct the error and generate a new datatape, rawinsonde data were obtained from NCAR. Fortunately, the NCAR data formats proved to be more suitable for the types of analyses performed in this study.

After new algorithms were written to process the NCAR database, and the database was downloaded to diskette, a series of preliminary profiles were generated. These revealed that the kinematic patterns were more coherent in the lower 7 km than above this level. Since winds in the 900 mb (1km) to 400 mb (7km) range are the most reliable, and the kinematically derived vertical velocity does not require the corrections specified by O'Brien (1970) in this region (Matthews, 1983), vertical profiles were restricted to the lower 7 km.

Most of the investigations were further narrowed to one quadrant of the rawinsonde network, because similar profiles were apparent in all quadrants. The southwest quadrant, bounded by Knowlton, Miles City, and Powderville (see Fig. 1), was selected because the first convective complex affecting the severe-storm period entered the mesonet from the southwest.

Data Adjustments

Interpolation

Rawinsonde data for specific times between soundings were synthesized using linear interpolation to account for varying sounding release times and differing balloon ascent rates, as in Fankhauser (1969). The interpolation scheme also provided a means to supply estimates for missing data. Using the

criteria outlined in Fuelberg and Printy (1983), interpolation for missing data was used only if the time interval between rawinsondes was less than 6 h, and the target time fell between the first and last observation times, thus avoiding extrapolation. Logarithmic interpolation between pressure levels was used when parameters were required for levels not corresponding to contact times.

Balloon Drift

In order to determine the degree that balloon drift affected the calculated kinematic values, divergence was calculated for the northwest and northeast quadrants of the rawinsonde network, with and without balloon drift (see Table 2). Since the values differed by less than ten percent in both cases, it was decided to ignore balloon drift in the algorithms.

Table 2. Divergence calculations with and without balloon drift at the 150 mb level on June 20, 1981 at 2200 GMT.

Quadrant	Divergence Values	
	Without Balloon Drift	With Balloon Drift
Northwest	$-3.82 \times 10^{-5} \text{ s}^{-1}$	$-3.50 \times 10^{-5} \text{ s}^{-1}$
Northeast	$-2.21 \times 10^{-4} \text{ s}^{-1}$	$-2.45 \times 10^{-4} \text{ s}^{-1}$

Surface/Upper-Air Comparisons

Network Differences

One of the factors affecting a network's ability to detect an atmospheric event is the observational sampling density. With a sampling distance of s , the minimum resolvable scale length is $2s$. This is analogous to the Nyquist frequency, which specifies that at a sampling period of T , the smallest period

that can be sampled is $2T$. Unfortunately, the possibility of interfering radio signals limits the spacing of the rawinsonde network to about 55 km (Fankhauser, 1969). The CCOPE rawinsonde spacing varied from about 60 to 85 km. The smallest event that could be sampled by the rawinsonde network, then, would have a scale length approximately three or four times that measurable by the surface network. This would seem to preclude a direct comparison between the two sampling networks (see Achtemeier, 1980). However, this problem was circumvented by using a limited set of surface stations, selected on the basis of their geographical locations relative to the rawinsonde release sites, in the comparison (MOQ, MUO, MUI in Fig. 1). Accordingly, the limited-station analysis (LSA) procedures developed by Heimbach and Engel (1986) for analyzing the CCOPE surface data were adapted for use with the rawinsonde data.

Coordinate Reference Frames

Surface parameters calculated by the LSA algorithms are based in a constant-altitude (AGL [Average Ground Level]) coordinate system. Therefore, the surface kinematic calculations fail to account for additional terms, i.e., wind shear and terrain slope, that appear in a non-horizontal reference frame (Schaefer, 1973). The rawinsonde calculations, however, are based in a constant-geopotential height above mean sea level (MSL) coordinate system. The analyses needed to allow for a comparison between the rawinsonde "surface" calculations, which would be at the horizontal level of the highest surface station, and the surface mesonet sites. To accomplish this, the rawinsonde programs calculated N "surface" parameters, where N is the number of stations at unique elevations. These calculations were "stepped-up" until the first possible horizontal level was attained. Thereafter, the parameters were

calculated at horizontal levels. The difference between the surface and first horizontal level, then, provided a measure of the effect of wind shear and terrain slope. Terrain-induced lifting, however, was not directly considered by the rawinsonde programs.

Data Processing Methods

All data processing was accomplished on an IBM PC, since the estimated costs to perform the analyses on a mainframe were prohibitive. Specialized software were developed for this study; however, a complete description of the programming procedures is beyond the scope of this paper. Therefore, documentation is limited to program listings, which are provided in the appendices. The PLOT-10 emulation software, used to generate the plots on a dot-matrix printer, is copyrighted and, therefore, not included.

Evaluation of Meteorological Parameters

The methods used by the LSA procedures to evaluate the meteorological parameters for the rawinsonde database are detailed below. Unless otherwise noted, the formulae follow Saucier (1972).

Vapor Pressure

The saturation vapor pressure, e_s , was calculated using the semi-empirical quadratic formulation given by Tabata (1973) :

$$\log e_s = 8.42926609 - 1.82717843(10^3/T) - 0.071208271(10^3/T)^2, \quad (1)$$

where e_s is in millibars, and T is the absolute temperature in degrees Kelvin.

The ambient vapor pressure, e , may be found by substituting the dewpoint temperature, T_d , for the ambient temperature in Eq. 1.

Absolute Humidity

The absolute humidity, or density of water vapor, ρ_w , is found using the equation of state :

$$\rho_w = m_w e / R^* T, \quad (2)$$

where m_w is the molecular weight of water (18 g/mol) and R^* is the universal gas constant (8.316963×10^7 erg/mol-deg).

Specific Humidity

The specific humidity, q , is the mass of water vapor per unit mass of the moist air mixture. The total density, ρ , of the air is the sum of the dry-air and the absolute humidity:

$$\rho = \rho_d + \rho_w = m_d (p - 0.37803e) / R^* T, \quad (3)$$

where m_d is the molecular weight of dry air (28.966 g/mol) and p is the ambient pressure in millibars. Thus, the specific humidity is given by :

$$q = \rho_w / \rho = 0.62197e / (p - 0.37803e). \quad (4)$$

Mixing Ratio

The mixing ratio, r , is the mass of water vapor per unit mass of the dry air with which it is mixed :

$$r = \rho_w / \rho_d = 0.62197e / (p - e). \quad (5)$$

Relative Humidity

The relative humidity, U , is the ratio of the actual to saturation mixing ratio at the same temperature and pressure, expressed as a percent :

$$U = (r/r_s) \times 100\%, \quad (6)$$

where r_s is the saturation mixing ratio.

Virtual Temperature

The virtual temperature, T^* , is the temperature of dry air having the same pressure and density of the moist air :

$$T^* = T(1 + 0.61r). \quad (7)$$

Potential Temperature

The potential temperature, θ , is the temperature of a parcel of air brought dry adiabatically to the 1000 mb level:

$$\theta = T(1000/p)^{(R_d/c_p)}, \quad (8)$$

where c_p is the specific heat of dry air at constant pressure (1.004×10^7 erg/g-deg), and R_d is the ideal gas constant for dry air (0.2870×10^7 erg/g-deg). The ratio R_d/c_p is 2/7 for dry air.

Equivalent Temperature

The equivalent temperature, T_E , is the temperature that a parcel of air would have if it were brought pseudo-adiabatically to the point where all latent heat is realized, and then returned dry-adiabatically to its original partial pressure (Rossby definition) :

$$T_E = T_e^{(Lr/c_p T)}, \quad (9)$$

where L is the latent heat of vaporization, 597.3 cal/g at 273 K.

Equivalent Potential Temperature

The equivalent potential temperature, θ_E , is the dry-adiabatic reduction of the equivalent temperature to the 1000 mb level:

$$\theta_E = T_E(1000/p)^{(2/7)}. \quad (10)$$

Total specific atmospheric energy

The total specific atmospheric energy, E_T , was determined using Darkow's (1968) formula, reduced to units of cal/g-s :

$$E_T = c_p T + gZ + Lq + V^2/2, \quad (11)$$

where Z is the MSL elevation. E_T essentially represents the "static" energy, since the kinetic energy term, $V^2/2$ is negligible.

Horizontal Kinematic Convergence

The horizontal kinematic convergence is the negative of the horizontal divergence of the velocity vector, and may be physically represented as the fractional areal expansion per unit time:

$$\text{Conv}_H = -\text{Div}_H = \vec{\nabla}_H \cdot \vec{V} \approx (1/A)(\Delta A/\Delta t). \quad (12)$$

The initial area bounded by the stations is A , and ΔA is the change in area due to the wind during the time interval Δt . The horizontal convergence was calculated using a special case of the line integral method which was analogous to the Bellamy (1948) technique. Details concerning the adaptation of the technique to computer processing are outlined in Heimbach and Engel (1986).

Vertical Component of the Relative Vorticity Vector

The vertical component of the relative vorticity vector (hereafter simply the relative vorticity) is $\vec{\nabla}_H \times \vec{V}_H$. This is equal to the divergence of the wind transformed by rotating the wind field clockwise 90 degrees:

$$\begin{aligned} u' &= v, \\ v' &= -u, \end{aligned} \quad (13)$$

where $()'$ indicates the transformed wind field.

Horizontal Moisture Convergence

The horizontal moisture convergence is estimated by multiplying the average absolute humidity, ρ , for the area by the horizontal kinematic convergence (Heimbach and Engel, 1986):

$$M_{con_H} = \rho_w \text{Conv}_H. \quad (14)$$

Vertical Velocity

The vertical velocity, w , is found by using the equation of mass continuity (see, e.g., Hess, 1959) :

$$\rho \nabla \cdot \mathbf{V} = 0, \quad (15)$$

and assuming an incompressible atmosphere ($d\rho = 0$) :

$$w = \int_0^H \text{Conv}_H \, dZ. \quad (16)$$

H is the elevation (MSL). The surface boundary condition has $w = 0$, as in Doneaud (1983). In the LSA programs, the vertical velocity was found by summing individual contributions :

$$w_N = \sum_{i=1}^N (1/2)(\text{Conv}_{H_{i-1}} + \text{Conv}_{H_i})(\Delta h) \quad (17)$$

where Δh is the change in height between selected levels.

CHAPTER 5

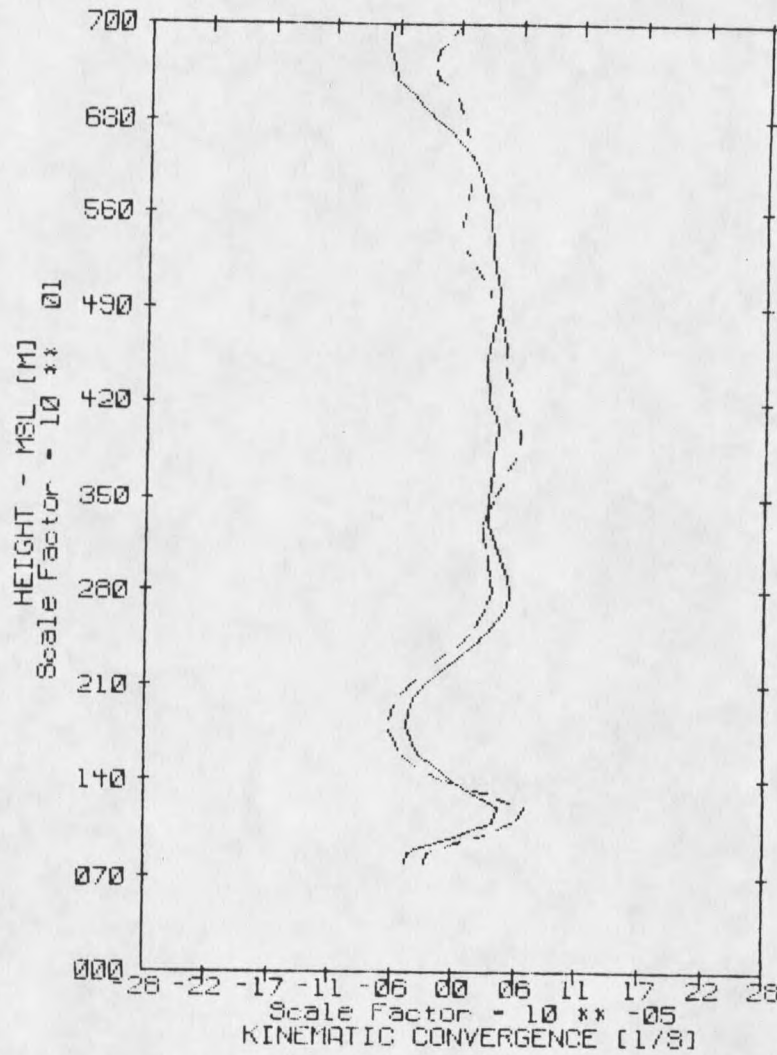
CASE-STUDY ANALYSES

The southwest quadrant of the rawinsonde network (bounded by Knowlton, Miles City, and Powderville) was the nearest to an approaching convective complex at the start of the severe-storm period (June 12-13). Therefore, it was selected as the primary quadrant for analysis. However, the center station in the rawinsonde network (Knowlton) had only one usable sounding during the HIPLEX-I period. As a result, the southern half of the network (bounded by Baker, Miles City, and Powderville) was selected for study during this period.

Knowlton lies on an approximately east-west line between Baker and Miles City, and is thus properly on the southern half boundary. Therefore, Knowlton should be included as a boundary station when the southern half is analyzed. Heimbach and Engel (1986), however, found that increasing the number of stations around the periphery of the area had little effect on the LSA calculations for the surface stations. This result appears to hold true for the upper-atmosphere as well. Figure 3 compares the horizontal convergence values at 2340 GMT on 12 June for the southern half of the rawinsonde network both with and without Knowlton. The differences are at the noise level. Similar results were observed for the other parameters tested.

The lack of sounding data at Knowlton during the HIPLEX-I period may also affect comparisons between the southwest surface quadrant and the upper-air, as well as comparisons among the case-study periods. However, Heimbach and Engel (1986) found that the responses of the derived parameters became smoother as

CV vs HM 0 01061223 40/ 40 GMT BAK MLS POW



CV vs HM 1 01061223 40/ 40 GMT B K M P

Figure 3. Kinematic convergence using 4 stations (dashed line) and 3 stations (solid line) about the southern half of the rawinsonde network at 2340 GMT on 12 June 1981.

the station spacing and averaging time increased, but that the major features were preserved. This was also seen to occur in the upper-air profiles. Figure 4 compares the relative vorticity profiles of the southwest quadrant and the southern half of the rawinsonde network at 2000 GMT on 12 June. The sinusoidal pattern has less amplitude for the southern half, but the waveforms parallel the southwest quadrant pattern. Similar trends were observed for the other parameters tested.

Since the trends on the upper-air profiles were not critically dependent on Knowlton, the lack of sounding data at this station on June 20 did not warrant abandoning this period for the present study. Additional support for this decision came from the one valid sounding that was available during this period. Patterns for the southwest quadrant for this sounding were consistent with those of the southern half.

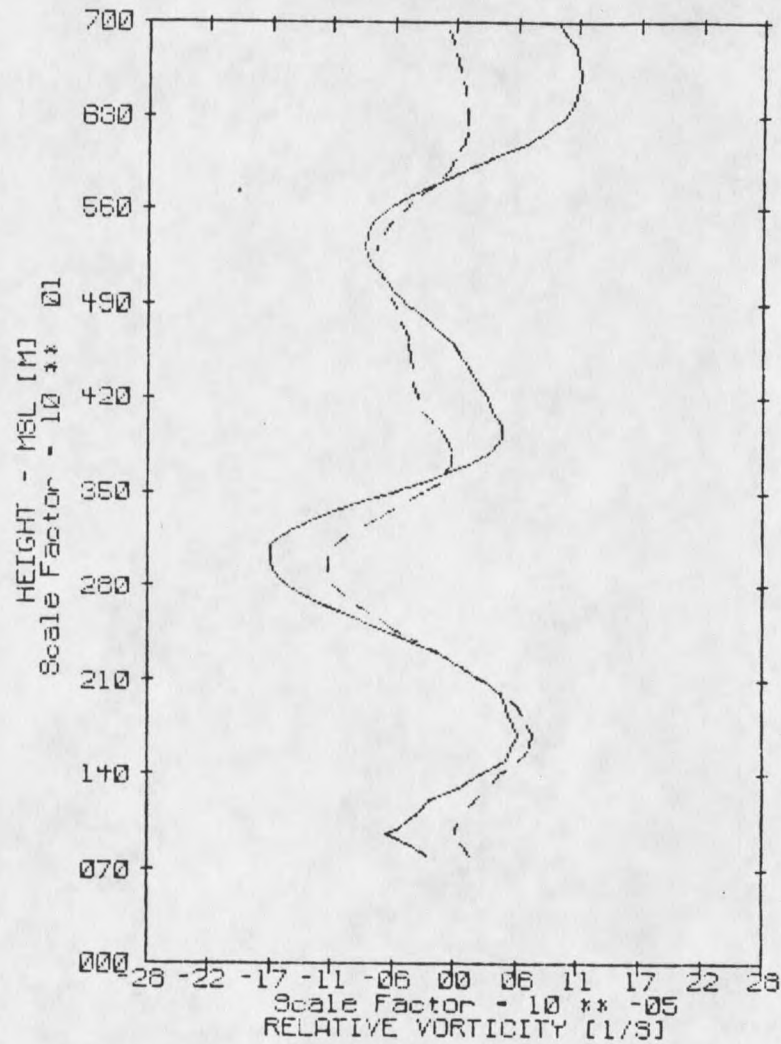
Case-Study : The Severe-storm Period (12-13 June 1981)

General Meteorological Conditions During the Severe-storm Case

The zonal flow evident earlier in the week at 500 mb (Fig. 5) gave way to mostly southwest flow over the CCOPE mesonet area by the afternoon of the 12th. The southwest flow preceeded a closed 500 mb low pressure system, which had moved eastward from a position just off the west coast at 1200 GMT (Greenwich Mean Time) on the 12th to a position in central Washington/Oregon by 1200 GMT on the 13th. A surface frontal system with its attendant low pressure center was located in northeastern Utah at 1200 GMT on the 12th, and it also moved eastward, arriving in northeastern Colorado after the 24-hour period.

The morning of the 12th had mostly cloudy skies and light, scattered rainshowers. A weak short-wave trough embedded in the upper level flow

RV vs HM 0 81061220 0/ 0 GMT KWL MLS POW



RV vs HM 1 81061220 0/ 0 GMT BAK ML3 POW

Figure 4. Relative vorticity for the southwest quadrant (solid line) and the southern half (dashed line) of the rawinsonde network at 2000 GMT on 12 June 1981.

FRIDAY, JUNE 12, 1981

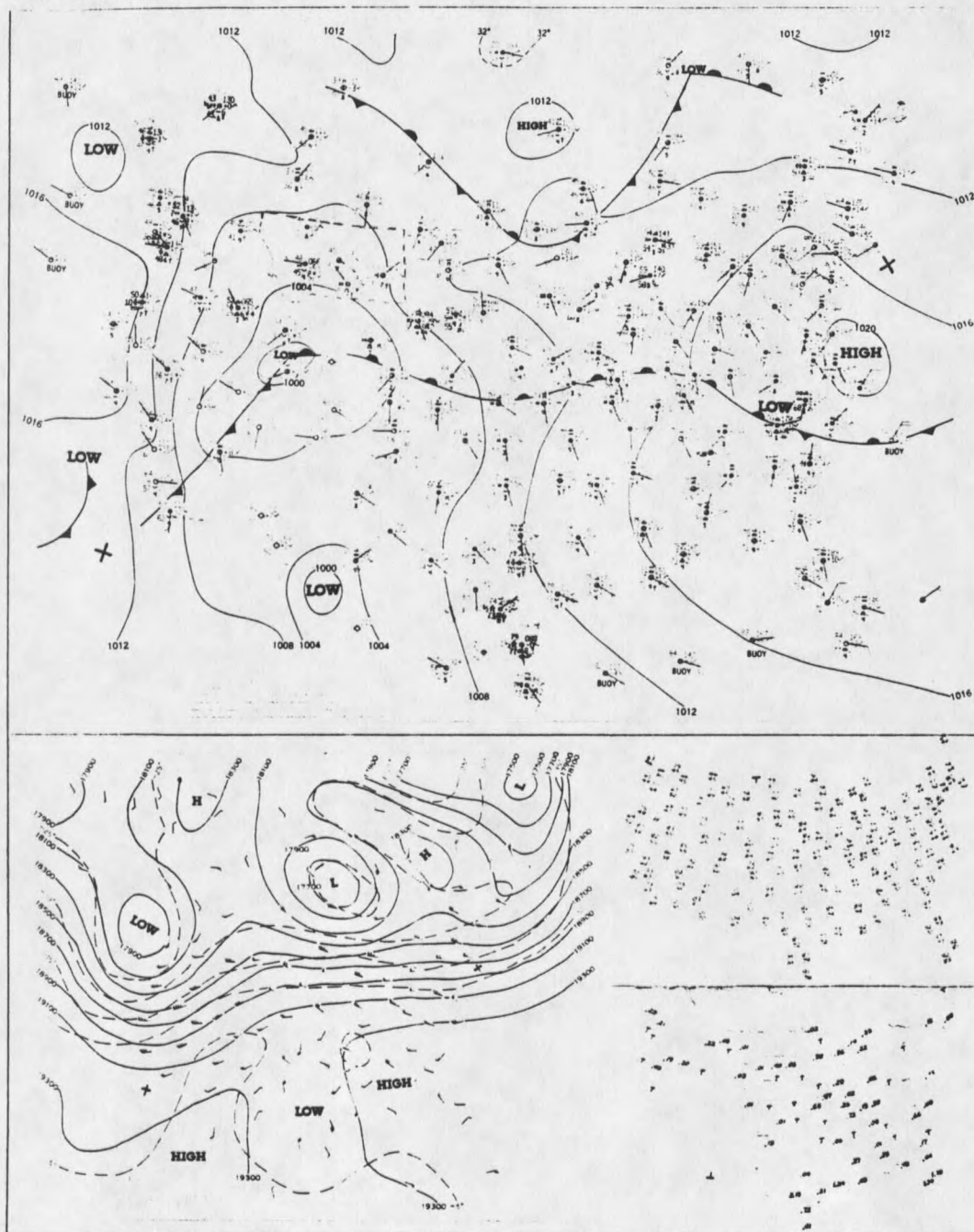


Figure 5. Surface and 500 mb synoptic maps at 1200 GMT on 12 June 1981, with 24 hour temperature extremes and precipitation amounts and areas (U.S. Dept of Commerce, 1981).

triggered some thunderstorm activity later in the day. A mature convective complex was first observed by the Skywater radar north-northwest of Miles City at 2200 GMT. Other complexes moved into the area from the southwest during the late afternoon.

Composite radar charts indicated possible light rain showers in the area bounded by the southwest quadrant of the rawinsonde network at 1600 GMT. However, this system had moved out of the southwest quadrant by 1800 GMT, and skies remained echo-free, but not cloud-free, until shortly after 0000 GMT, when a system approached the area from the southwest. This northeast-moving system had cloud tops to 13 km AGL, with radar reflectivities exceeding 50 dBz, indicating probable hail (Hartzel, 1981). By 0230 GMT, the system had moved out of the southwest quadrant. Field notes remarked that a tornado watch had been put in effect for the area south of Miles City.

Surface Patterns During the Severe-storm Case

Figure 6 shows the precipitation trace for the severe-storm period. The period covers 12-13 June; but the figure's abscissa (time) continually increases rather than resetting to zero at hour 24. In order to facilitate reference to these figures, times will be specified in terms of the abscissa labels, e.g., 2515 refers to 0115 GMT on the 13th. All surface plots represent averaged or derived parameters using three stations (MOQ, MUO, MUI).

Three distinct precipitation events are evident: a small event, from 2430 to 2515; an intense moderate event at 3000, followed by a light rainfall afterward; and a large event from 3215 to 3545. These are discussed separately below.

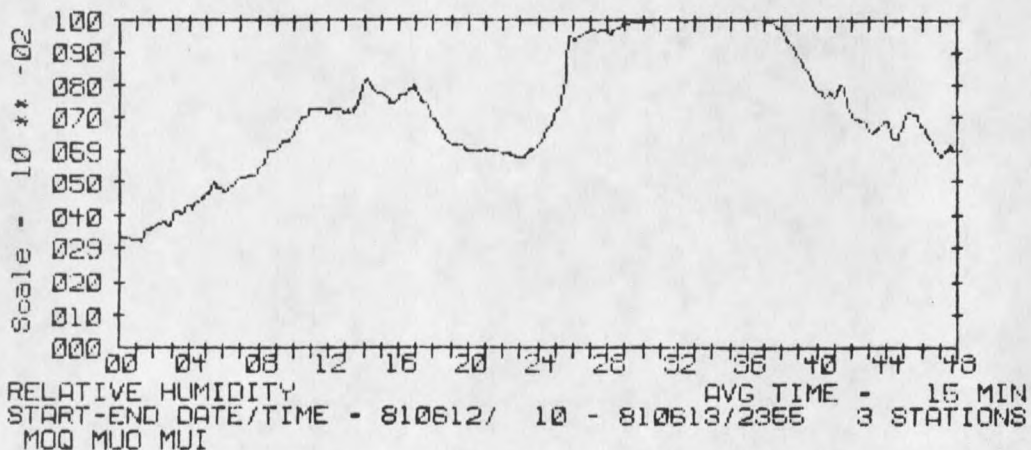
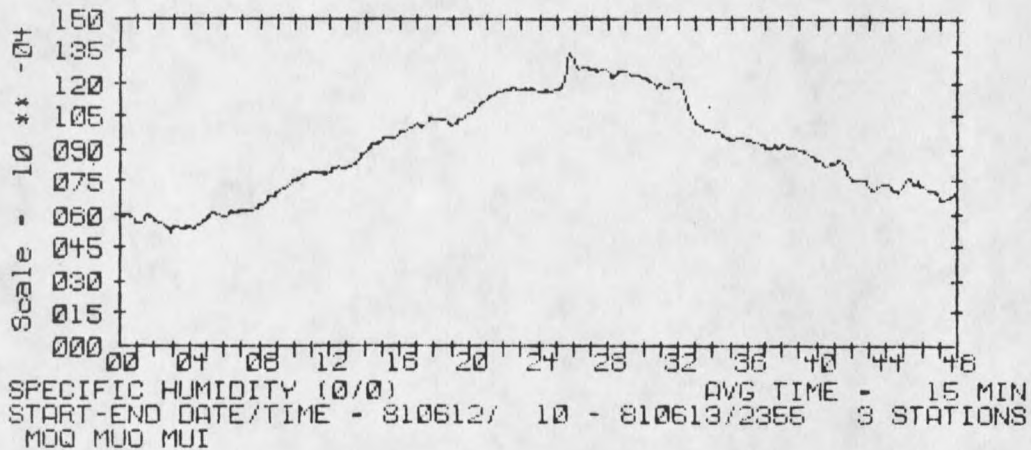
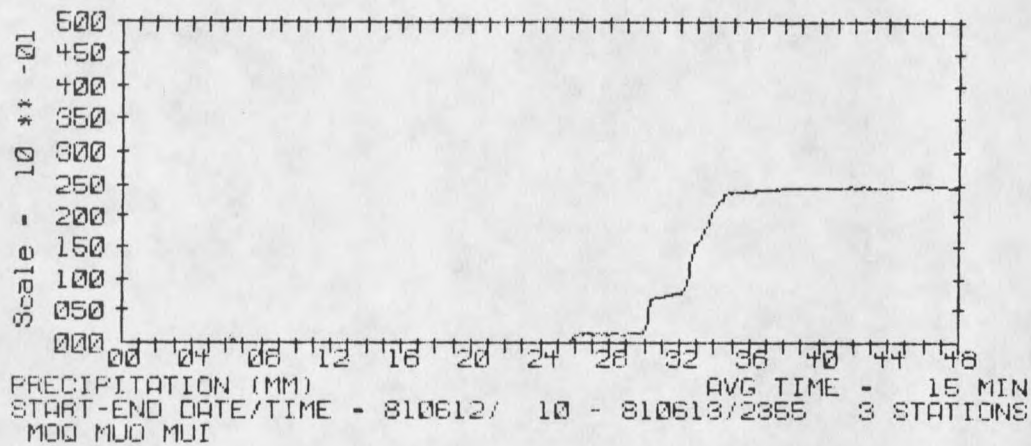


Figure 6. Precipitation, specific humidity, and relative humidity for the southwest surface quadrant bounded by MOQ, MUO, and MUI on 12-13 June 1981.

Surface Indications Prior to Precipitation Event #1. The specific humidity trace (Fig. 6) showed a steady increase in surface water until 2130, at which time it leveled off. The increase in surface water supply was due to the advection of maritime-tropical air from the Gulf of Mexico. During the same period, the relative humidity (Fig. 6) paralleled the specific humidity trace until 1700, then started to decrease in response to the rising temperature (Fig. 7). It continued its decrease until 2000, when it became more steady. The total specific atmospheric energy (Fig. 7) showed an increase until 2230, after which it began to decrease.

The surface winds (Fig. 8), generally from the east-southeast, gradually increased in speed until 1930, when a decrease occurred. The surface moisture convergence and kinematic convergence (Fig. 9) were mostly positive until 1915, at which time an alternating convergence/divergence pattern was established which persisted until 2315, when the kinematic and moisture convergence showed the largest increase of the day. Relative vorticity (Fig. 9) was mostly negative during the alternating convergence patterns.

Surface Indications During Precipitation Event #1. A small pressure jump (Fig. 7) at 2430 reversed the falling pressure pattern prevalent earlier in the day, and accompanied the onset of rain. The wind speed increased slightly, also reversing a trend of falling wind speeds, but the wind direction remained essentially constant. At the time of the onset of rain, or shortly thereafter, the surface moisture convergence peaked, and began its largest drop of the day (to that time). After the rain had stopped (2515), the specific humidity showed a sharp increase, probably as a result of evaporation of the newly fallen precipitation. The relative humidity remained high throughout the period.

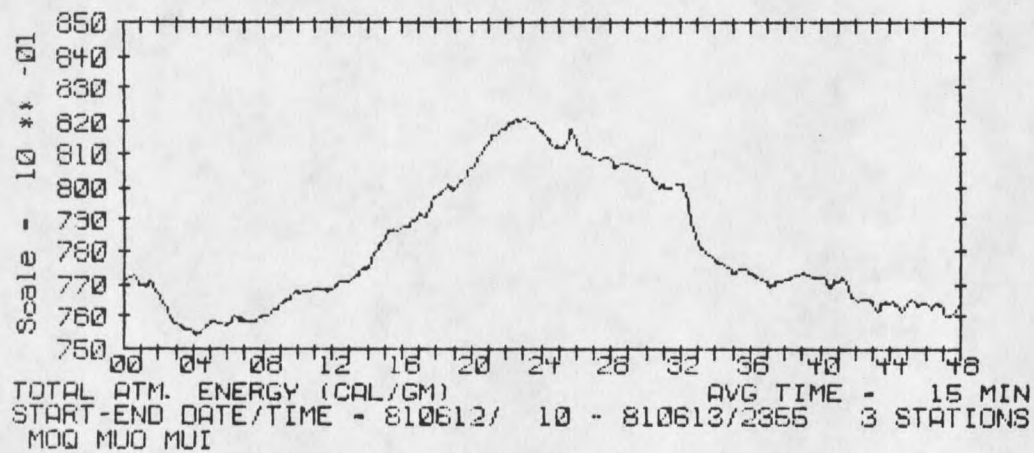
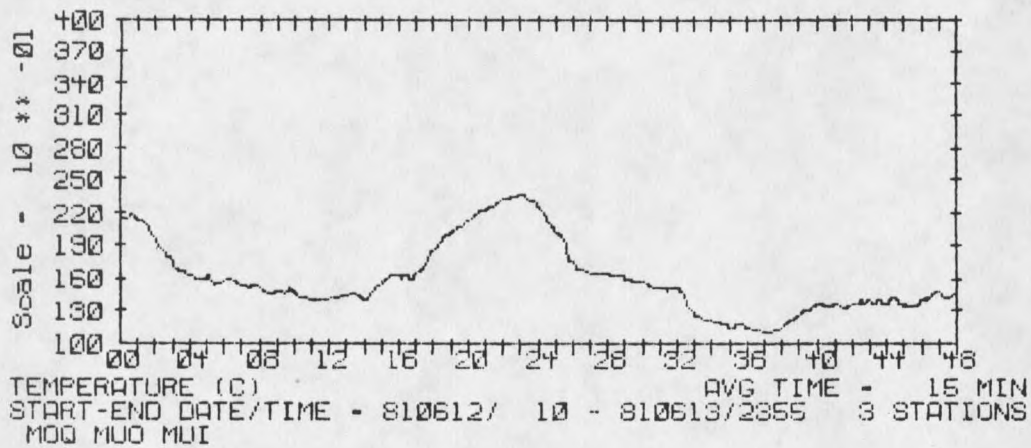
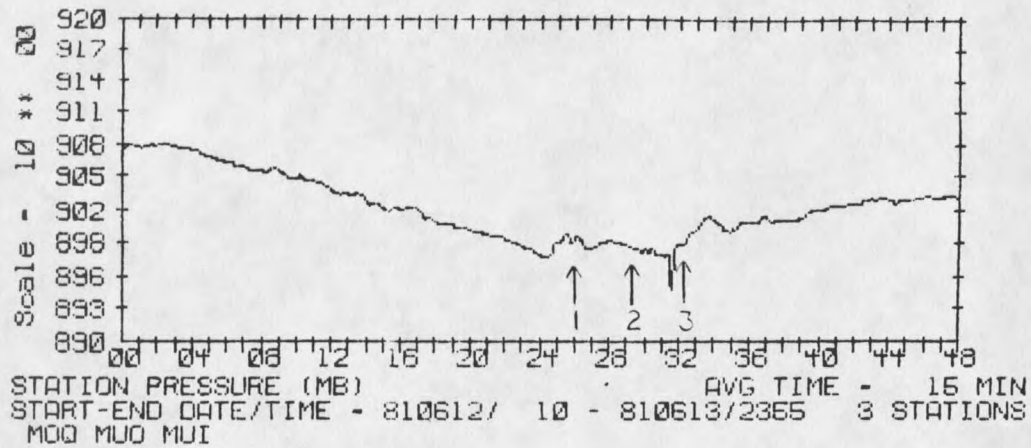


Figure 7. Station pressure, ambient temperature, and total specific atmospheric energy for the southwest surface quadrant bounded by MOQ, MUO, and MUI on 12-13 June 1981.

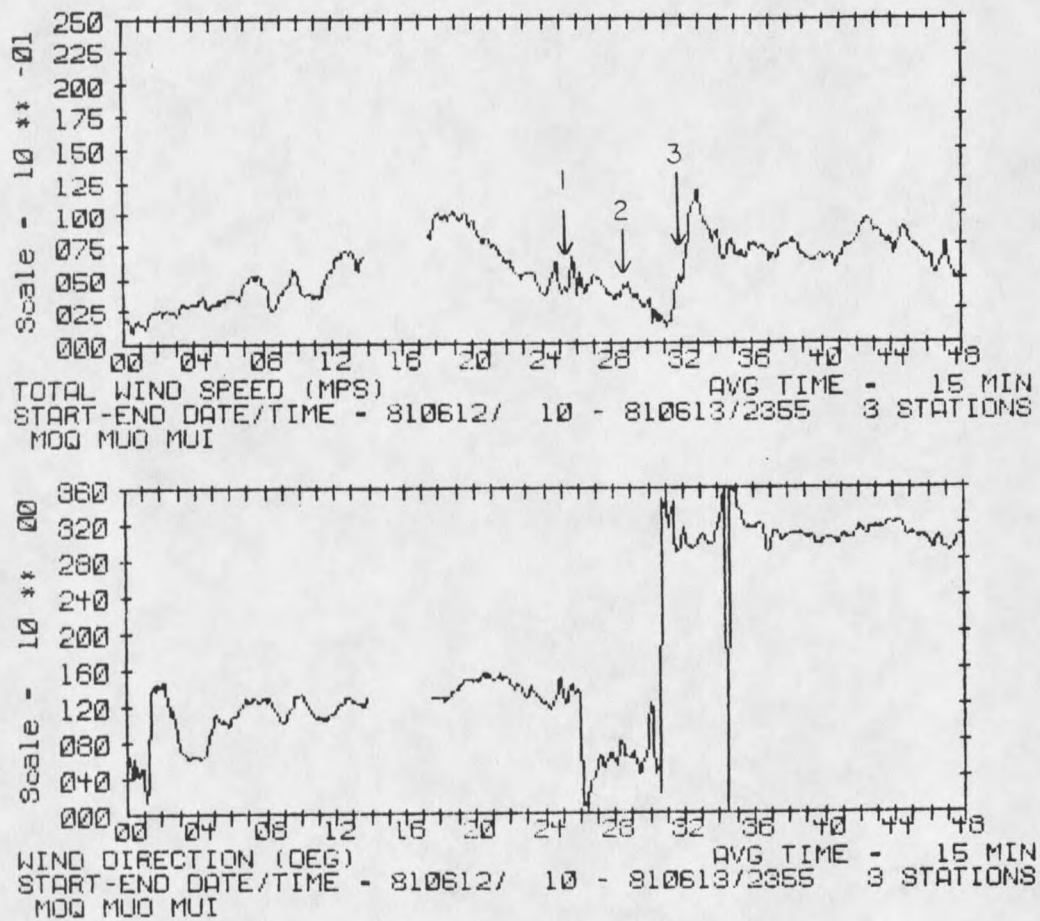


Figure 8. Wind speed and wind direction for the southwest surface quadrant bounded by MOQ, MUO, and MUI on 12-13 June 1981.

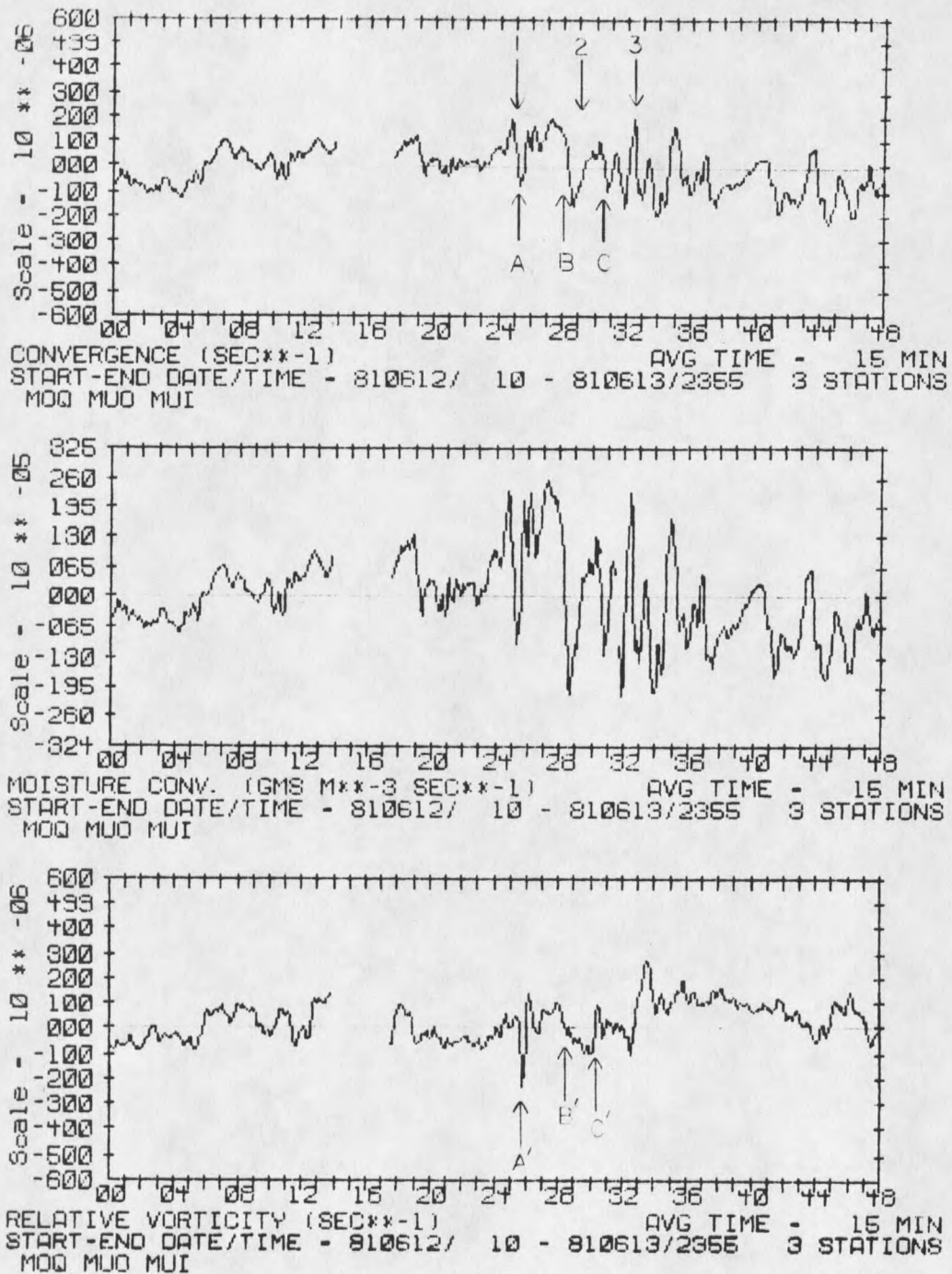


Figure 9. Kinematic convergence, moisture convergence, and relative vorticity for the southwest surface quadrant bounded by MOQ, MUO, and MUI on 12-13 June 1981.

The moisture convergence reached a relative minimum value at about the time the rain stopped, and rapidly rose to its pre-rain, positive values. The moisture convergence trace remained positive until 2715, after which it began a pattern of alternating positive and negative values until approximately 3700. No precipitation accompanied the moisture convergence drop at 2715, its largest magnitude change during the day. At 2600, the wind shifted, becoming more easterly.

Surface Indications During Precipitation Event #2. A short, intense burst of precipitation at 3000 was followed by a light rainfall until 3215. Though initially more intense than the first event, the second event was not accompanied by a pressure jump. The wind direction did shift to the northwest shortly after the start of precipitation (3030), but the wind speed continued its falling trend. The moisture convergence, continuing its alternating positive/negative trend, did start dropping when the rain started, reaching its peak negative value shortly after the cessation of the intense rain. The relative vorticity started an alternating positive/negative trend, becoming cyclonic when surface convergence was positive, and anticyclonic when divergence occurred.

Surface Indications During Precipitation Event #3. This event was preceded by the largest pressure change of the day. At 3130, a sharp drop in pressure was followed almost immediately by a sharp rise. The onset of rain brought a sharp temperature drop which was accompanied by a corresponding drop in the specific humidity since the relative humidity was already at 100 percent. The wind speed increased, and, after peaking at approximately 3300, remained relatively steady thereafter. During this period, the moisture convergence was

alternating signs at approximately 45 min intervals. The peak positive moisture convergence value occurred approximately 15 min after the onset of rain, and continued its alternating pattern throughout the event. The relative vorticity continued the pattern established during event #2, until the termination of rain, after which it remained cyclonic.

Summary of Major Surface Features During the Severe Storm Case. Prior to the convective rainfall, the surface pressure steadily dropped, probably in response to an upper-level, short-wave trough. During this time, advected moisture increased the specific humidity, and the surface kinematic and moisture convergence showed a slight oscillatory pattern, though the values remained mostly positive. Relative vorticity was mostly negative during the pre-storm period.

About an hour before the onset of rain, the kinematic and moisture convergence showed a significant increase, and the relative vorticity became less anticyclonic. After the first convective event, the kinematically derived parameters displayed larger amplitude oscillations, with horizontal convergence coupled with cyclonic relative vorticity. After the final precipitation event, the relative vorticity remained positive, but the convergence, while continuing its oscillatory nature, was mostly negative. The surface pressure slowly increased, and the specific humidity decreased.

Upper-Air Profiles During the Severe-storm Case

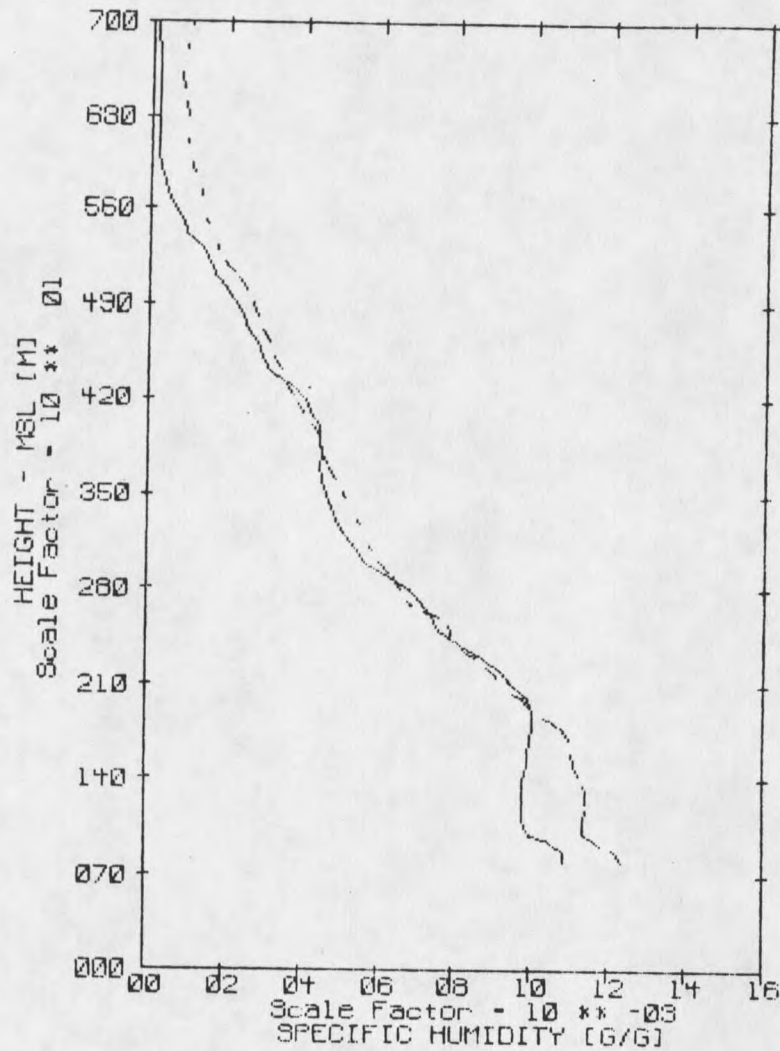
Rawinsondes were taken at approximately 90 min intervals from 1930 GMT to 2340 GMT, thus covering the pre-storm environment. Unfortunately, no sounding information was available during the precipitation events associated with the convective complexes.

The specific humidity profiles for the lower 7 km of the atmosphere in the southwest quadrant of the rawinsonde network at 2000 and 2340 GMT are shown in Fig. 10. A significant increase (about 1.3 g/kg) in the water content of the lower 2 km occurred during this period, with a lesser increase above 4.2 km. The advection of this low-level moisture, from layer theory considerations, has the effect of destabilizing the atmosphere (see, e.g., Newton, 1963). The nearly vertical profile from the first non-intersecting horizontal level (about 1 km) to 1.4 km is consistent with Betts' (1973) observation of a well-mixed water vapor profile throughout the mixing layer in the pre-rain environment.

The relative humidity in the lower 4.3 km, however, remained essentially unchanged (Fig. 11), due to the temperature increase (Fig. 12) in this layer. The relative humidity increase above this level is due to the increased water supply, since the temperature remained essentially constant in this upper layer. The equivalent potential temperature profile (Fig. 13) is nearly constant near the surface, as observed in the subcloud region of the inflow zone by Fankhauser (1982). The increase of about 4 K in the equivalent potential temperature in the lowest 2 km from 2000 to 2340 GMT is a response to both the increased temperature and the increased water content in this layer.

The total specific atmospheric energy profile (Fig. 14) depicts a mostly stable layer below 2 km for both periods, with an unstable layer above. A stable layer aloft (3.0 - 4.2 km) at 2000 GMT, which could enhance the opportunity for precipitation by trapping low-level moisture (Newton, 1963; Mahrt, 1980), had destabilized by 2340 GMT, resulting in a mostly unstable regime above 2 km to a height of nearly 7 km. However, the shallow, stable layer from 2.5 to 2.7 km, evident also in the equivalent potential temperature

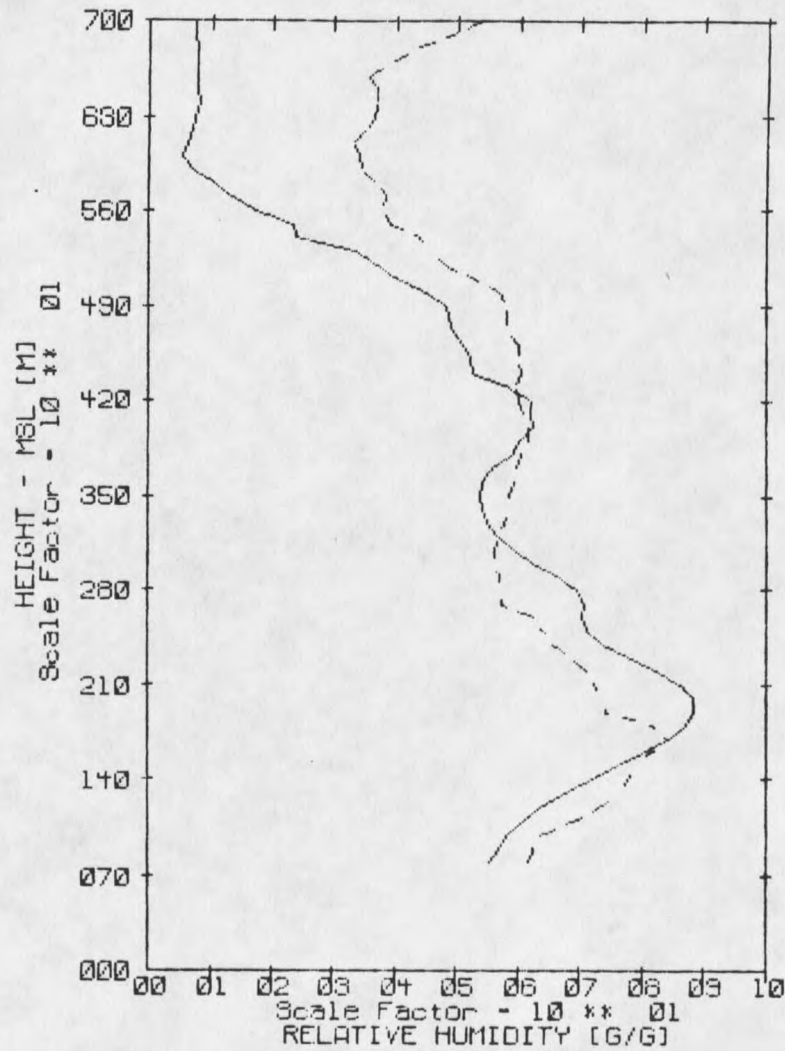
SH vs HM 0 81061220 07/ 0 GMT KWL MLS POW



SH vs HM 1 81061223 40/ 40 GMT KWL MLS POW

Figure 10. Specific humidity at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 June 1981 for the southwest quadrant of the rawinsonde network.

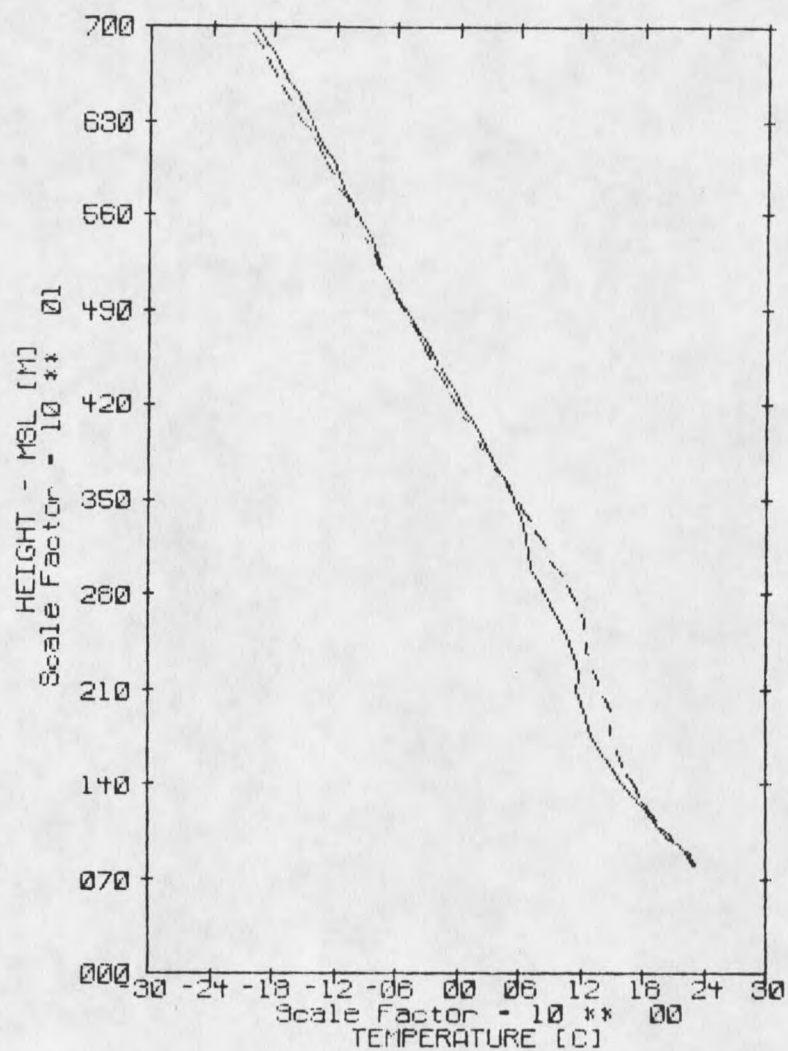
RH vs HM 0 81061220 0/ 0 GMT KWL MLS POW



RH vs HM 1 81061223 40/ 40 GMT KWL MLS POW

Figure 11. Relative humidity at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 June 1981 for the southwest quadrant of the rawinsonde network.

TE vs HM 0 81061220 0/ 0 GMT KWL MLS POW



TE vs HM 1 81061223 40/ 40 GMT KWL MLS POW

Figure 12. Ambient temperature at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 June 1981 for the southwest quadrant of the rawinsonde network.

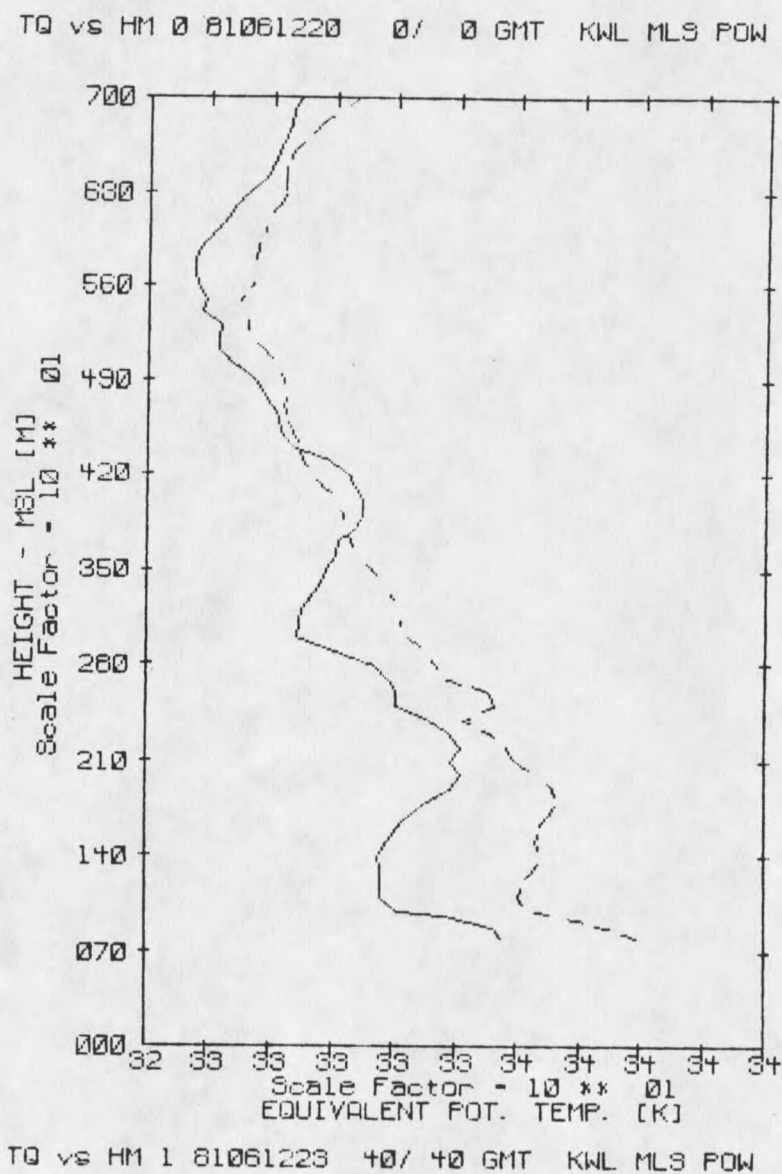
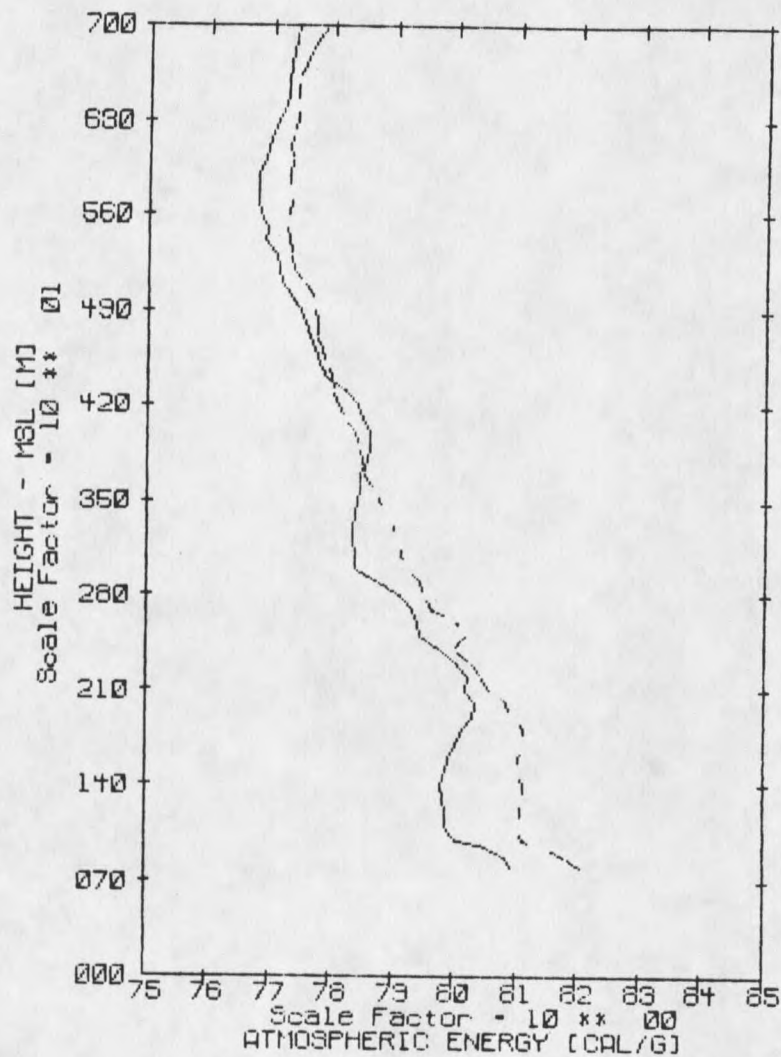


Figure 13. Equivalent potential temperature at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 June 1981 for the southwest quadrant of the rawinsonde network.

AE vs HM 0 81061220 0/ 0 GMT KWL MLS POW



AE vs HM 1 81061223 40/ 40 GMT KWL MLS POW

Figure 14. Total specific atmospheric energy at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 June 1981 for the southwest quadrant of the rawinsonde network.

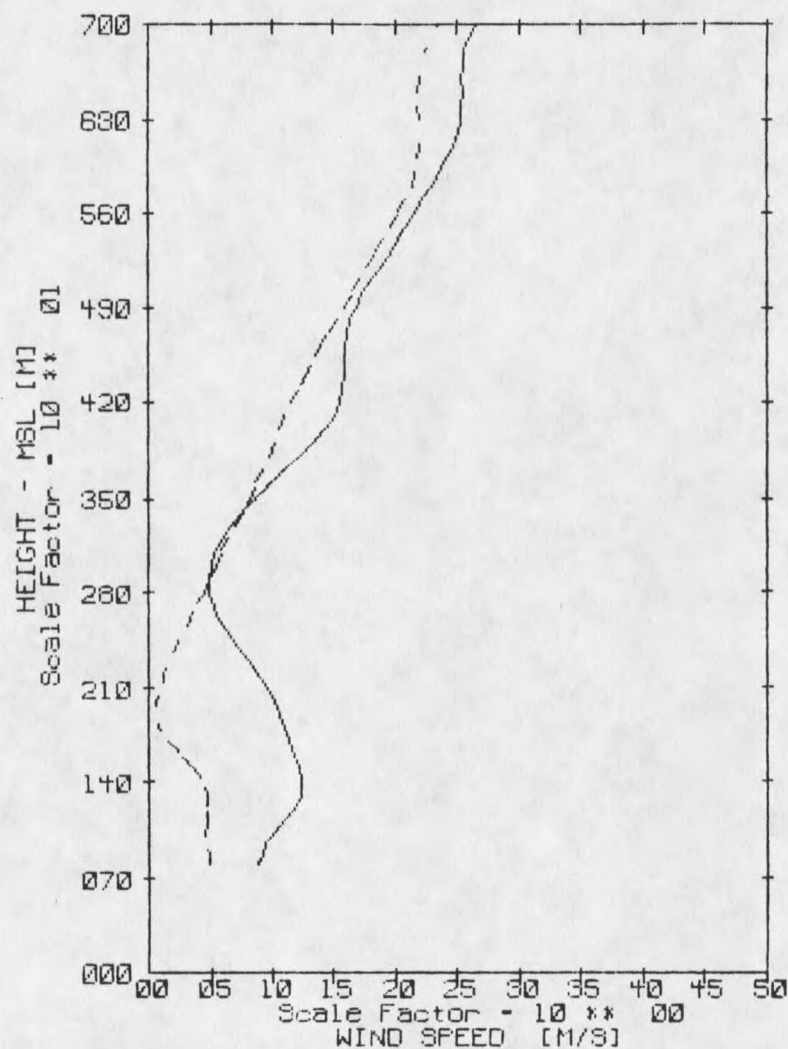
profile, but not in the ambient temperature trace (Fig. 12), apparently represents a concentration of moisture at this level -- a result not readily noticeable in the specific humidity profile alone.

A low-level jet was evident in the wind speed profile (Fig. 15), at the 1.4 km level at 2000 GMT. By 2340 GMT, the low-level jet had disappeared, and the wind speed minimum evident at 3.3 km at 2000 GMT had dropped to 2 km. The wind speed became nearly constant below this level. During the same period, a gradual wind-direction shear in the 1.4 - 4.2 km layer (Fig. 16) was replaced by a sharp wind-direction shear in the 1.9 - 2.2 km layer.

The vertical velocity (Fig. 17), was essentially zero throughout the lower 7 km at 2000 GMT. However, an updraft above 3.3 km, which steadily increased with height, was evident by 2340 GMT. This is a result of an alternating horizontal kinematic convergence/divergence pattern (Fig. 18) at 2000 GMT changing to a profile dominated by convergence above 2.2 km by 2340 GMT (see Eq. 17). The convergence-dominated patterns above 2.2 km did not change the general sinusoidal nature of the profile. This is even more evident in the moisture convergence trace (Fig. 19), which shows an enhanced amplitude over the kinematic convergence because of the increased water vapor content of the lower atmosphere.

An interesting feature of the convergence profiles is the earthward propagation of these waves. That these waves do propagate downward is evident by observing that the peculiar "kink" in the wave at 3.1 km (point A in Fig. 19) has been preserved for over 3.5 h, being evident at 1.8 km (point A' in Fig. 19), even though the maximum amplitude of the wave changed from $-1.98 \times 10^{-4} \text{ g/m}^3\text{-s}$ to $-1.34 \times 10^{-4} \text{ g/m}^3\text{-s}$. Since this kink is not evident in the kinematic convergence trace, it must be a result of a downward moisture flux.

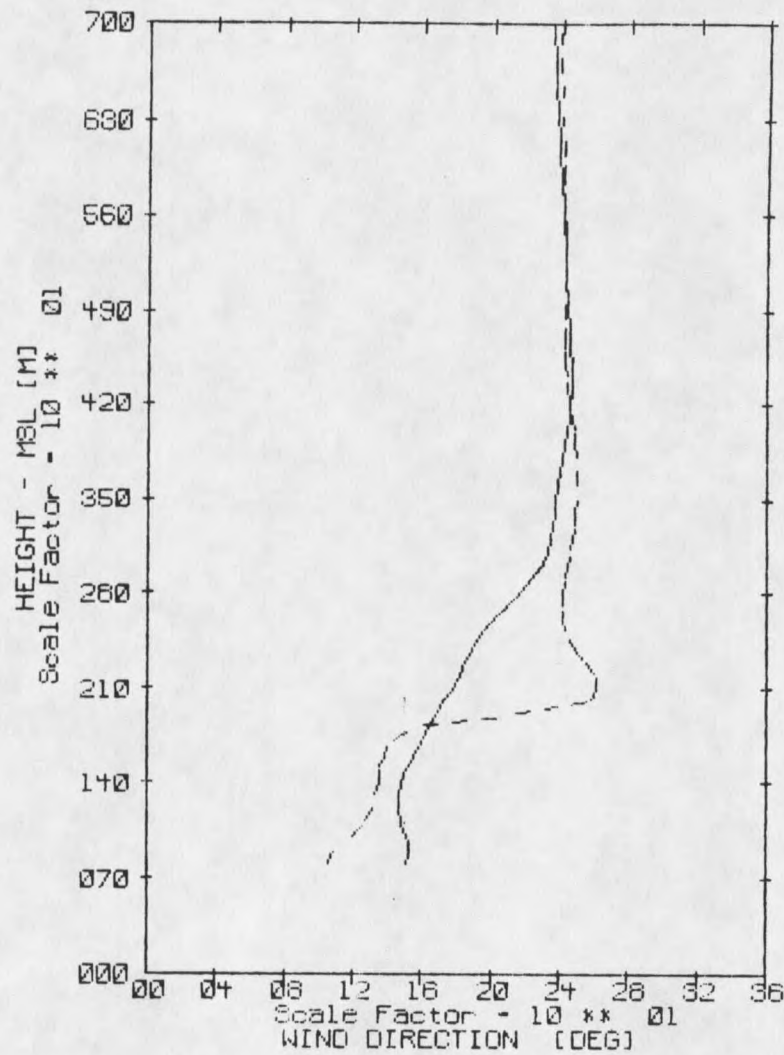
WS vs HM 0 81061220 0/ 0 GMT KWL MLS POW



WS vs HM 1 81061223 40/ 40 GMT KWL MLS POW

Figure 15. Wind speed at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 June 1981 for the southwest quadrant of the rawinsonde network.

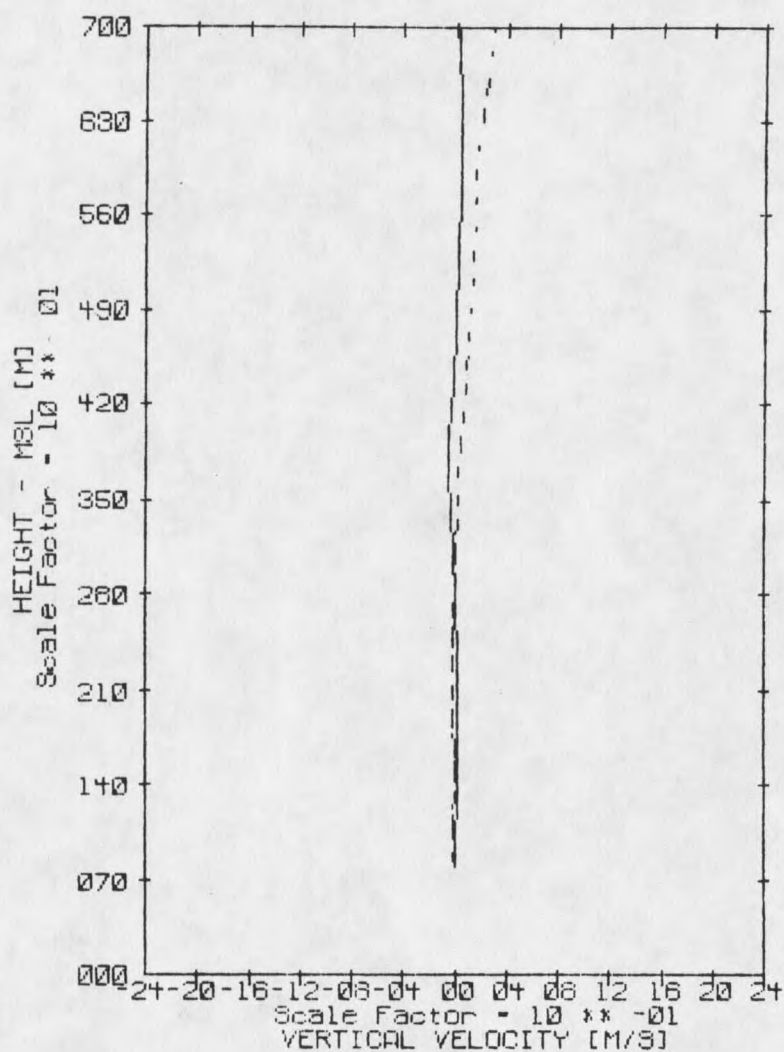
WD vs HM 0 81061220 0/ 0 GMT KWL MLS POW



WD vs HM 1 81061223 40/ 40 GMT KWL MLS POW

Figure 16. Wind direction at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 June 1981 for the southwest quadrant of the rawinsonde network.

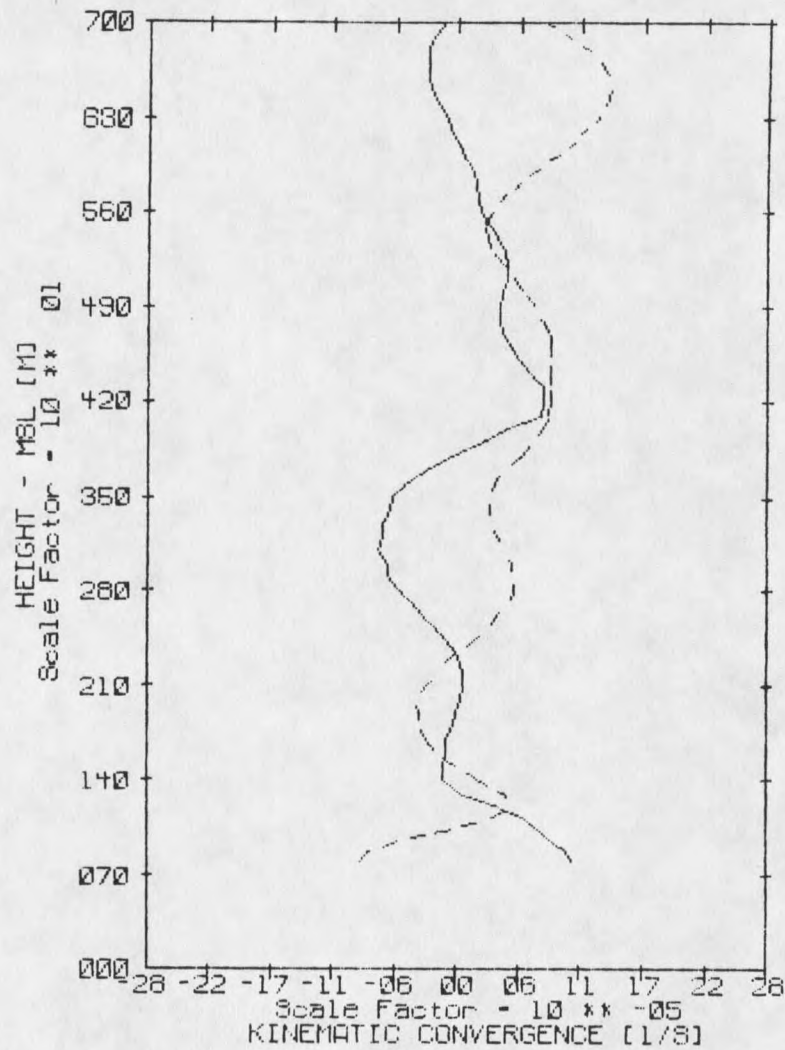
WW vs HM 0 81061220 07 0 GMT KWL ML3 POW



WW vs HM 1 81061223 40/ 40 GMT KWL ML3 POW

Figure 17. Vertical velocity at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 June 1981 for the southwest quadrant of the rawinsonde network.

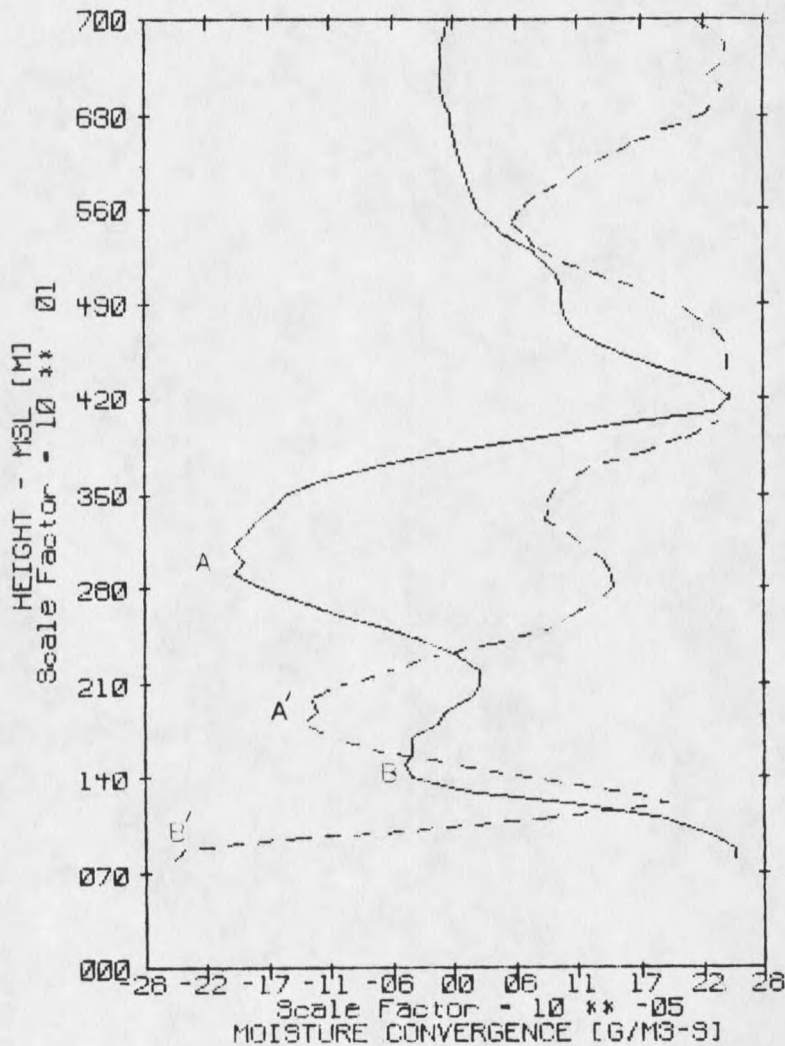
CV vs HM 0 81061220 0/ 0 GMT KWL MLS POW



CV vs HM 1 81061223 40/ 40 GMT KWL MLS POW

Figure 18. Kinematic convergence at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 June 1981 for the southwest quadrant of the rawinsonde network.

MC vs HM 0 81061220 0/ 0 GMT KWL ML3 POW



MC vs HM 1 81061223 40/ 40 GMT KWL ML3 POW

Figure 19. Moisture convergence at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 June 1981 for the southwest quadrant of the rawinsonde network.

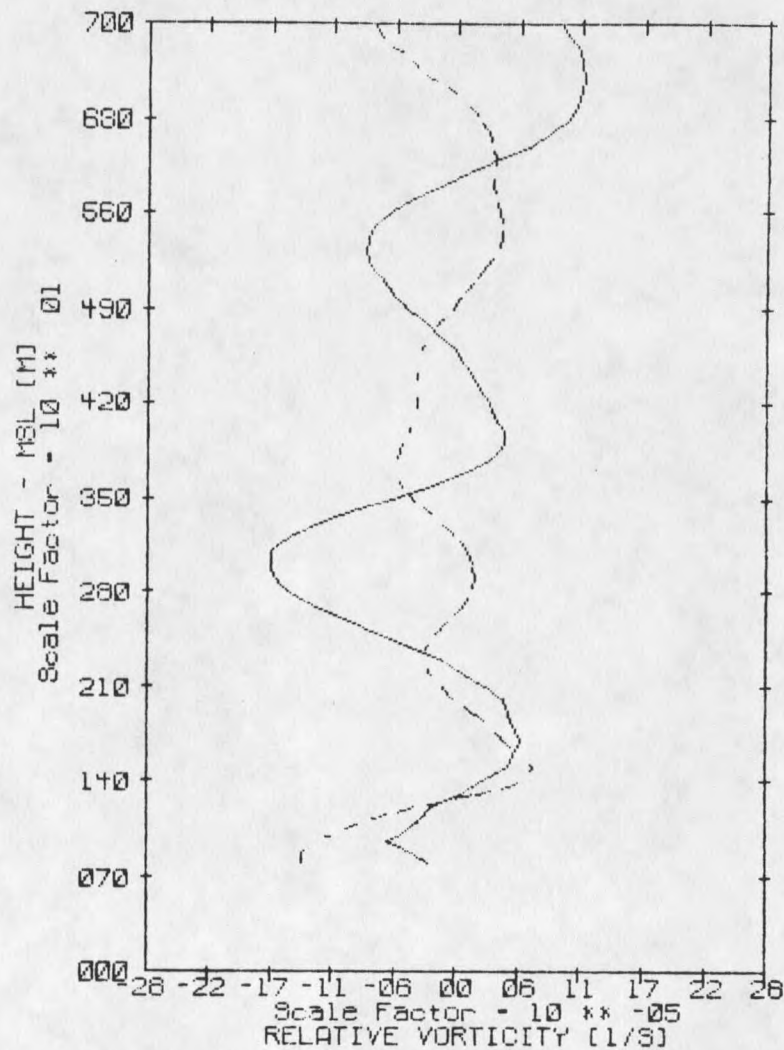
The descent of mesocyclones aloft has recently been observed by radar (Milner, 1986).

The relative vorticity also shows a vertical sinusoidal pattern which also appears to propagate downward (Fig. 20). However, the relationship between kinematic convergence and relative vorticity is difficult to ascertain. In some cases they are of equal sign, in others of opposite sign. At 2000 GMT, (Fig. 21), the surface divergence is attended by cyclonic relative vorticity. In the layer above, the convergence is associated with anticyclonic relative vorticity, to 2.5 km. However, by 2340 GMT, the convergence and vorticity have become more coupled in this layer, sharing the same sign, but lagging by approximately $1/2$ wavelength (Fig. 22). This is analagous to the "checkerboard pattern" of surface convergence and relative vorticity observed by Heimbach and Engel (1986).

Surface/Upper-Air Coupling During the Severe-storm Case

Evidence for surface/upper-air coupling may be found by identifying a feature on the upper-air profiles which is also apparent on the surface pattern. The surface response would not be expected to occur simultaneously with the upper-air event; since a finite time would be required for the upper-air feature to propagate downward, or for the surface feature to work upward. An analogy would be a long rope which is fixed at one end, with a pertubation initiated at the other. A sinusoidal wave will be observed, which will propagate toward the fixed end. The rope here serves as the medium through which the pertubation is transferred, "coupling" the events at one end of the rope to the other. A short time later, the initial wave (damped somewhat) is observed at the fixed end. Of course, the atmosphere is a three-dimensional

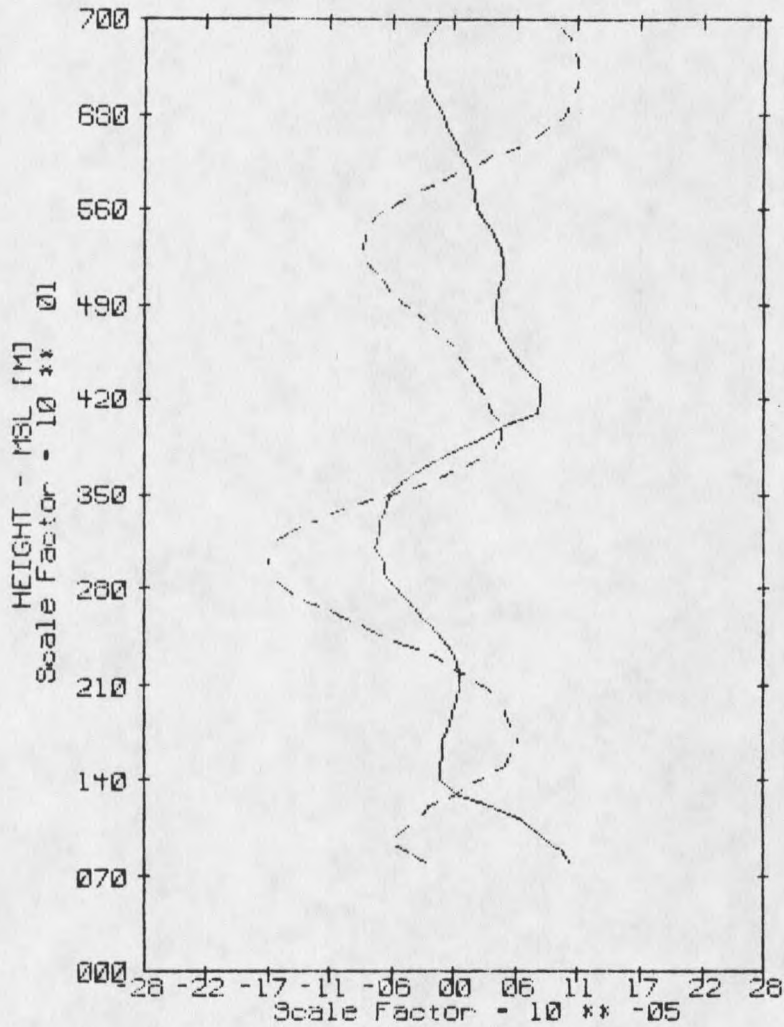
RV vs HM 0 81061220 0/ 0 GMT KWL MLS POW



RV vs HM 1 81061223 40/ 40 GMT KWL MLS POW

Figure 20. Relative vorticity at 2000 GMT (solid line) and 2340 GMT (dashed line) on 12 June 1981 for the southwest quadrant of the rawinsonde network.

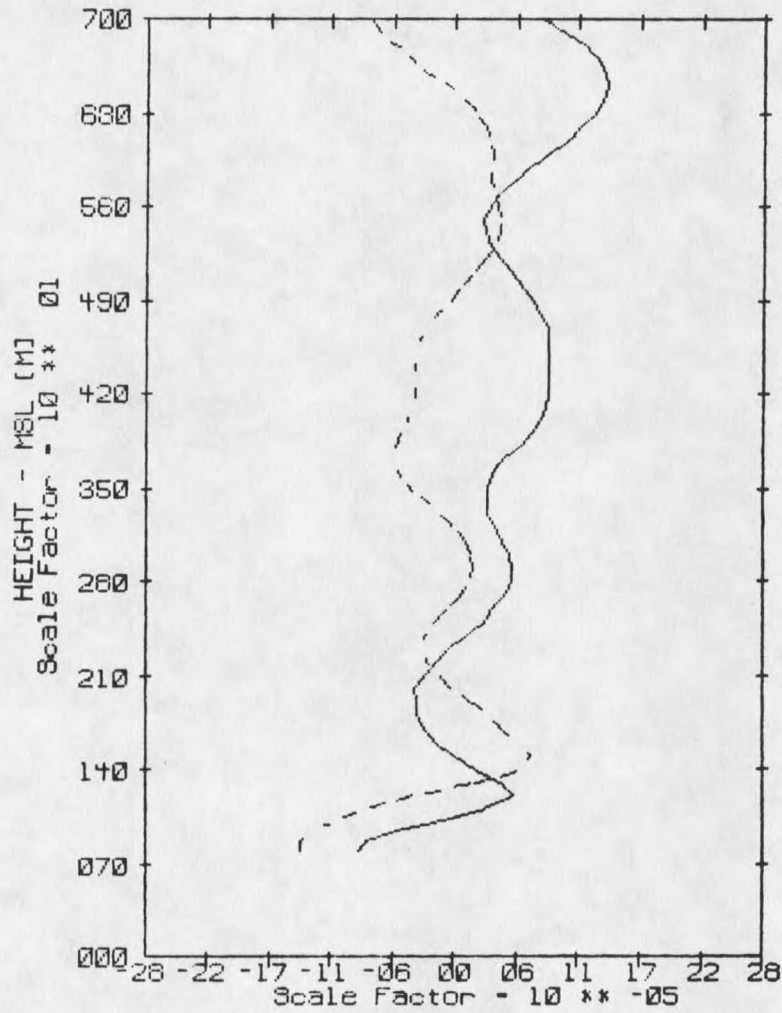
CV vs HM 0 81061220 0/ 0 GMT KWL ML3 POW



RV vs HM 1 81061220 0/ 0 GMT KWL ML3 POW

Figure 21. Kinematic convergence (solid line) and relative vorticity (dashed line) at 2000 GMT on 12 June 1981 for the southwest quadrant of the rawinsonde network.

CV vs HM 0 81061223 40/ 40 GMT KWL MLS POW



RV vs HM 1 81061223 40/ 40 GMT KWL MLS POW

Figure 22. Kinematic convergence (solid line) and relative vorticity (dashed line) at 2340 GMT on 12 June 1981 for the southwest quadrant of the rawinsonde network.

fluid (not a two-dimensional solid); thus, evidence of coupling may be more difficult to ascertain.

On the afternoon of the 12th, the vertical sinusoidal pattern evident in the kinematic parameters persisted throughout the sampling period. By 2340 GMT, the lower moisture convergence maximum had intensified and moved closer to the surface. The surface trace generally reflects this pattern, showing low values throughout the period, with a general increase in the moisture convergence toward the end of the period. The clear evidence of the downward propagating vertical waves suggests that these features will eventually be seen in the surface patterns; however, a point by point comparison with the surface trace shows a rather poor agreement between simultaneous upper-air and surface values. For example, Table 3 shows a surface kinematic convergence value of 0.4 s^{-1} for the stations MOQ, MUO, and MUI with a corresponding rawinsonde surface value of 1.02 s^{-1} . Similarly poor comparisons are evident at higher levels, and for the relative vorticity and moisture convergence.

Table 3. Point Comparisons Between Upper-Air and Surface Kinematic Parameters for 12 June 1981.

Parameter	Time ¹	Surface ²	Upper-Air						
			0m	1000m	1100m	1200m	1300m	1400m	1500m
Kinematic Convergence ³	2000	0.4	1.02	0.79	0.63	0.35	0.04	-0.11	-0.12
	2340	0.5	-0.85	-0.22	0.24	0.53	0.41	0.18	-0.01
Relative Vorticity ³	2000	-0.3	-0.24	-0.50	-0.34	-0.24	0.04	0.23	0.45
	2340	-0.3	-1.36	-1.14	-0.81	-0.37	0.17	0.64	0.69
Moisture Convergence ⁴	2000	0.4	2.75	2.21	1.85	1.08	0.13	-0.38	-0.44
	2340	0.7	-2.54	-0.68	0.80	1.88	1.53	0.68	-0.03

¹GMT

²MOQ, MUO, MUI

³ 10^{-4} s^{-1}

⁴ $10^{-4} \text{ g/m}^3 \text{ s}$

At least three factors contribute to the inconsistencies between the surface mesonet values and the surface calculated (0 - 1000 m) rawinsonde values :

1. The surface values consist of 15 min averages, but the rawinsonde measurements represent instantaneous values. The degree of disagreement between the two would depend on the magnitude of the variance in the wind speed and wind direction, i.e., the turbulence.

2. The rawinsonde release points were purposely chosen in somewhat sheltered areas to stabilize the balloon's fluctuations during ascent through the lower levels. The wind in these areas may be affected by the terrain features at the site, and may not necessarily represent the exposed surface wind.

3. Friction and other boundary-layer effects become more dominant closer to the surface. Thus, kinematic features in the upper-atmosphere that maintain their value for a significant portion of their descent could be significantly modified upon interaction with the boundary layer. In the rope example, the wave would reflect from the fixed end, and interfere with any subsequently arriving pulses.

The above factors don't preclude a comparison between the upper-air and the surface. Rather than a direct point-by-point comparison between surface and upper-air values, coupling may be shown by comparing patterns between the surface and upper-air.

The surface convergence patterns possess a rough sinusoidal pattern (Fig. 9), similar to that observed in other studies (see, e.g., Cooper, et al., 1982; Holle and Watson, 1983; Watson and Blanchard, 1984). The sinusoidal surface pattern would be expected if the vertical waves continued their earthward

propagation. That these upper waves do propagate through the surface layer is evident in Fig. 19. The wavefront at B at 2000 GMT has reached the surface (B') by 2340 GMT. Interaction with the surface layer had significantly modified its value, however, from $-0.44 \times 10^{-4} \text{ g/m}^3\text{-s}$ to $-2.54 \times 10^{-4} \text{ g/m}^3\text{-s}$.

It is unfortunate that the soundings were terminated prior to the passage of the convective complex. Based on the profiles obtained in the pre-storm environment, however, it would be expected that the upper vertical waves would increase in amplitude and perhaps frequency during the convective events occurring later in the period.

Case-Study : The HIPLEX-I Period (20-21 June 1981)

General Meteorological Conditions During the HIPLEX-I Case

The June 20 synoptic analysis produced a relatively featureless surface map under the influence of a primarily zonal flow at 500 mb (Fig. 23). This flow had been established earlier in the week and continued throughout the afternoon. Attendant with the flow aloft were embedded short-wave troughs causing sufficient instability to support HIPLEX-I type convection. Composite radar charts indicated that a small complex approached the area from the southwest at 2200 GMT, and progressed through the southwest quadrant by 0000 GMT, heading northeast. Light, scattered showers during the 24 h period resulted in .02 inches of precipitation at the Miles City FAA.

Surface Patterns During the HIPLEX-I Case

The relative minimum in the specific humidity at 2200 (Fig. 24) opposes the trend evident during the severe-storm period, which had shown a steadily increasing pattern during the same period. The surface water vapor

SATURDAY, JUNE 20, 1981

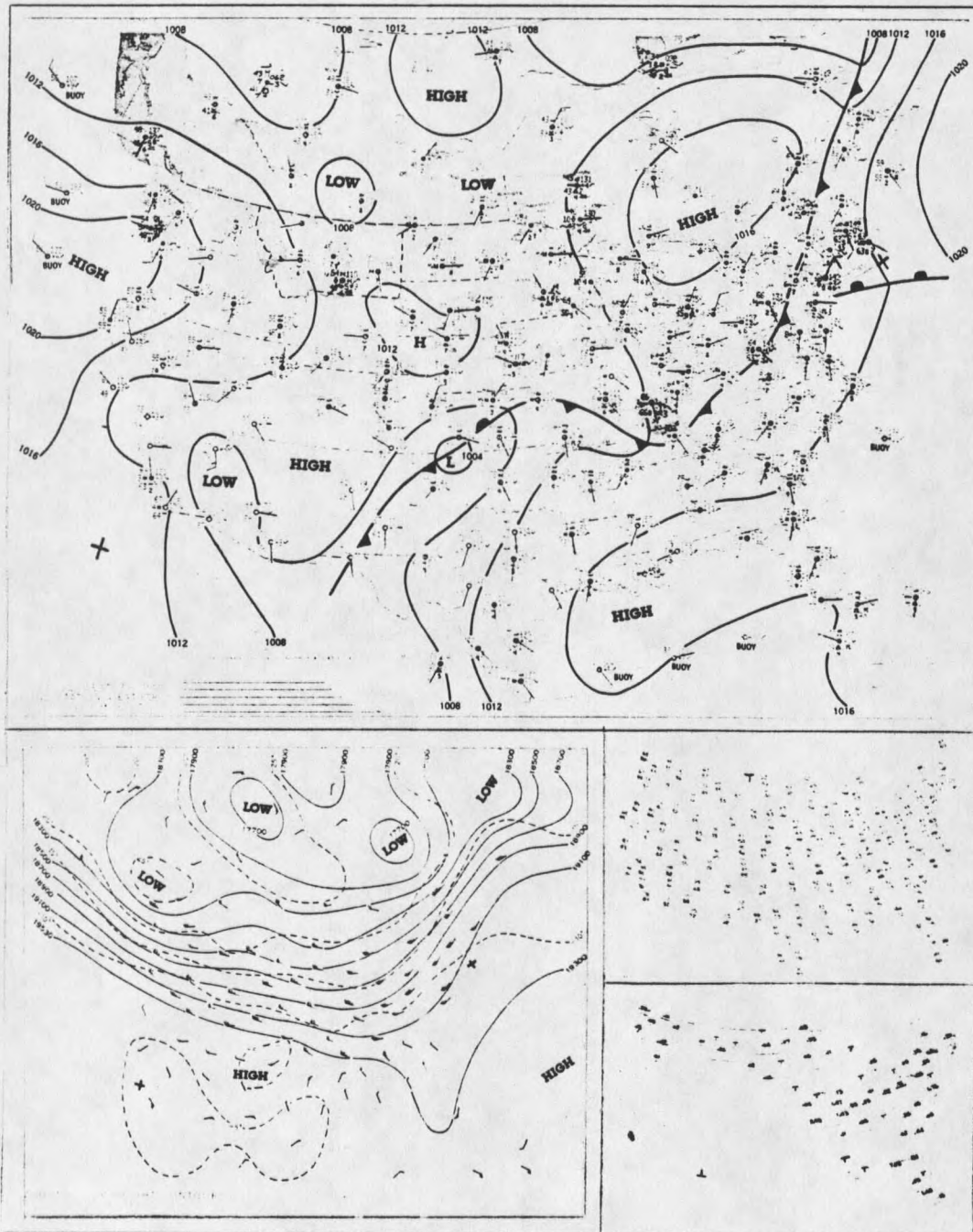


Figure 23. Surface and 500 mb synoptic maps at 1200 GMT on 20 June 1981, with 24 hour temperature extremes and precipitation amounts and areas (U.S. Dept. of Commerce, 1981).

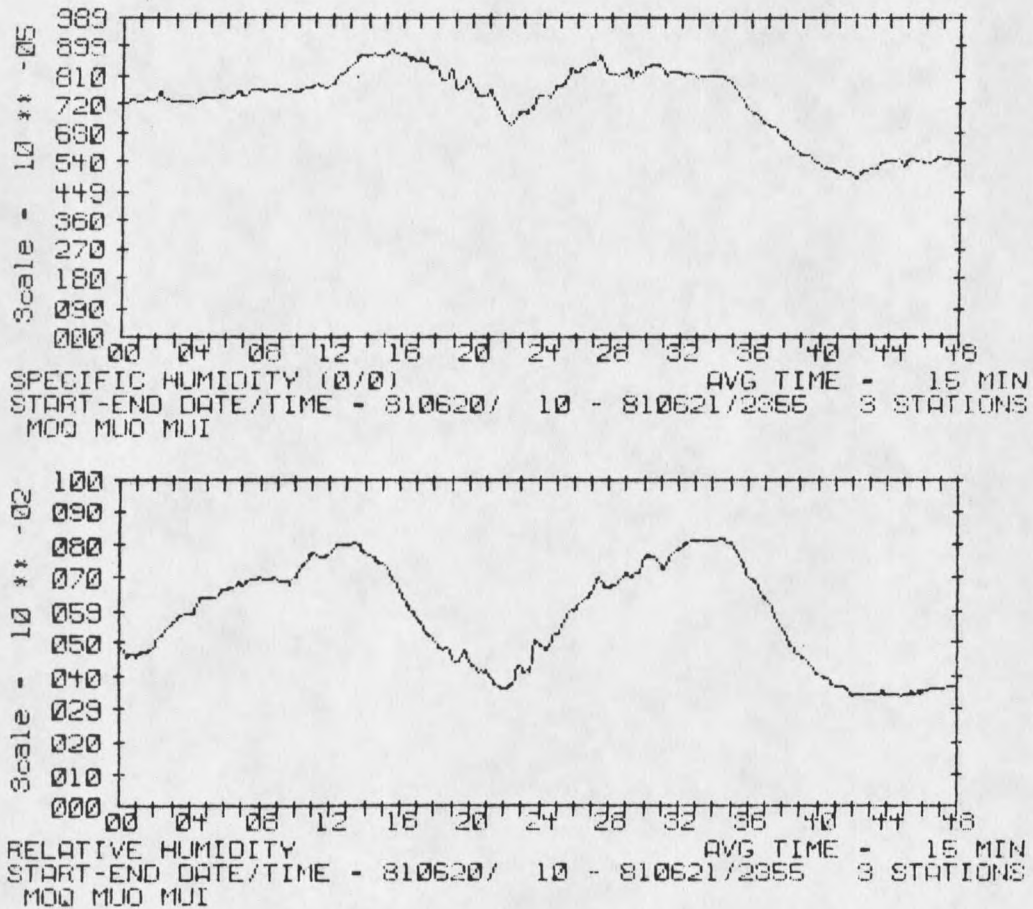


Figure 24. Specific humidity and relative humidity for the southwest surface quadrant bounded by MOQ, MUO, and MUI on 20-21 June 1981.

thereafter, remaining high until 3430, after which it again decreased. The relative humidity (Fig. 24) trace parallels the diurnal temperature pattern (Fig. 25). The total specific atmospheric energy (Fig. 25), though possessing essentially the same trend as that observed during the severe-storm period, is approximately 4 cal/g lower than the afternoon of the 12th.

The kinematic and moisture convergence (Fig. 26) were mostly positive until 1900, after which an alternating, sinusoidal divergence/convergence regime was established which persisted until 2900. During this time, the relative vorticity stayed mostly positive, although it also displayed a sinusoidal pattern.

Upper-Air Profiles During the HIPLEX-I Case

Rawinsondes were taken at approximately 90 min intervals from 1900 GMT to 0030 GMT. However, missing winds at Knowlton during the 1920 and 2109 GMT, and missing humidity data at Knowlton during the 2233 GMT sounding combined to virtually eliminate Knowlton from consideration, and effectively eliminated any quadrant analysis. Further, the lack of wind information above 700 mb at Baker for the 1910 GMT sounding delayed the analysis of the southern half of the rawinsonde network until 2130 GMT.

A slight increase in the specific humidity occurred at most levels during the interval from 2130 - 2344 GMT (Fig. 27). The atmospheric energy (Fig. 28), though increasing slightly during the same period, is only slightly unstable at 2130 GMT, and becomes more stable by 2344 GMT. The wind speed and wind direction (not shown) remained essentially unchanged throughout the period, and lacked the low-level jet and direction shear evident during the severe-storm period. The vertical velocity (Fig. 29) increased with time above 2 km.

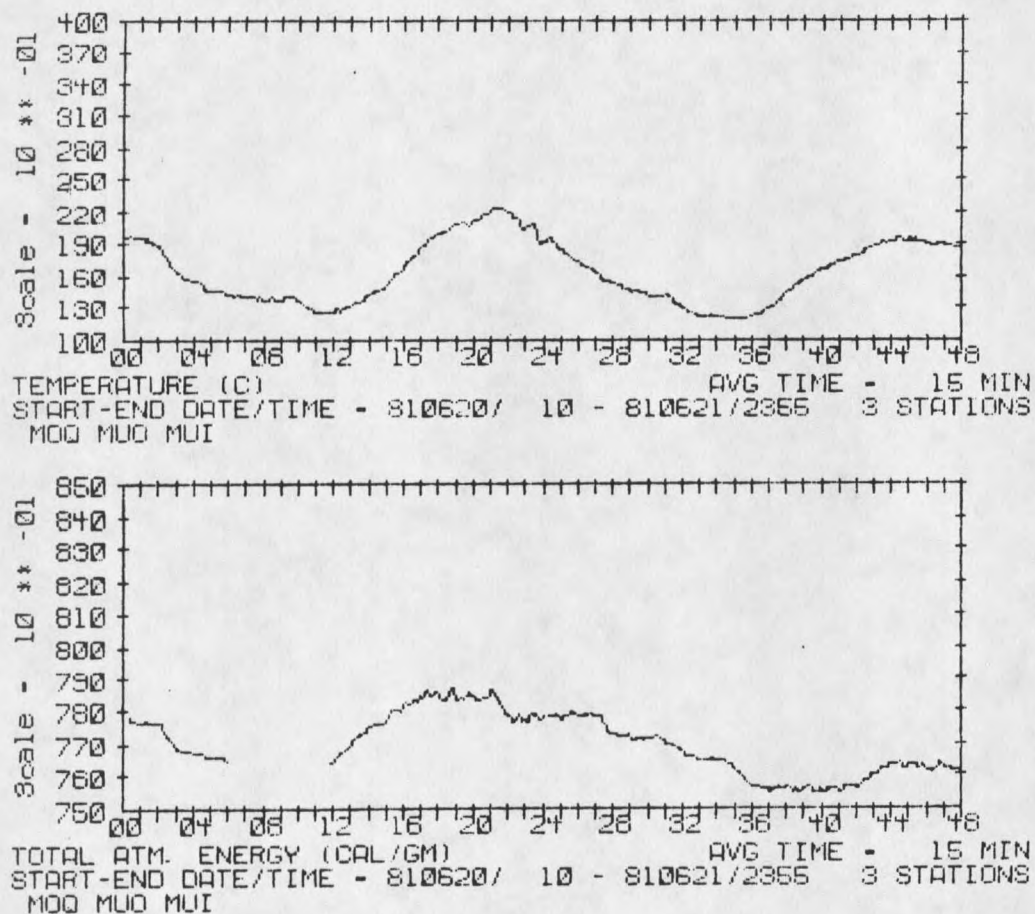


Figure 25. Ambient temperature and total specific atmospheric energy for the southwest surface quadrant bounded by MOQ, MUO, and MUI on 20-21 June 1981.

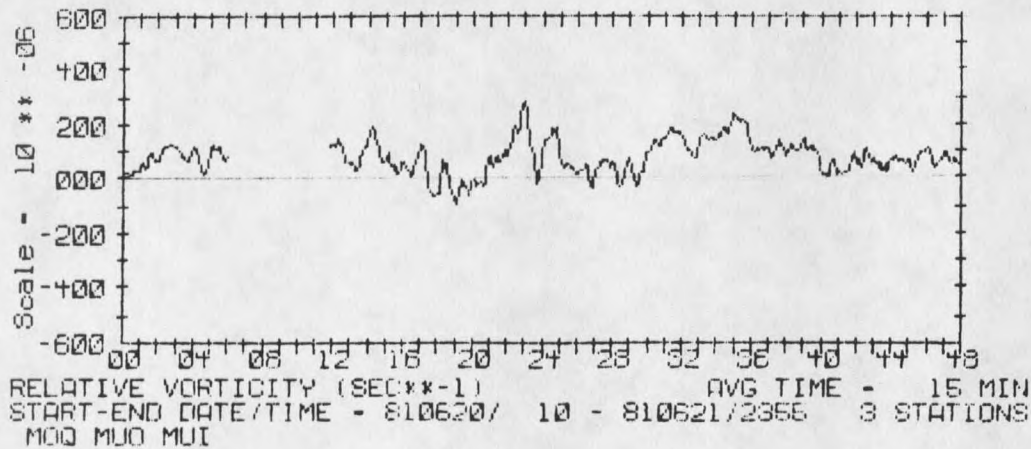
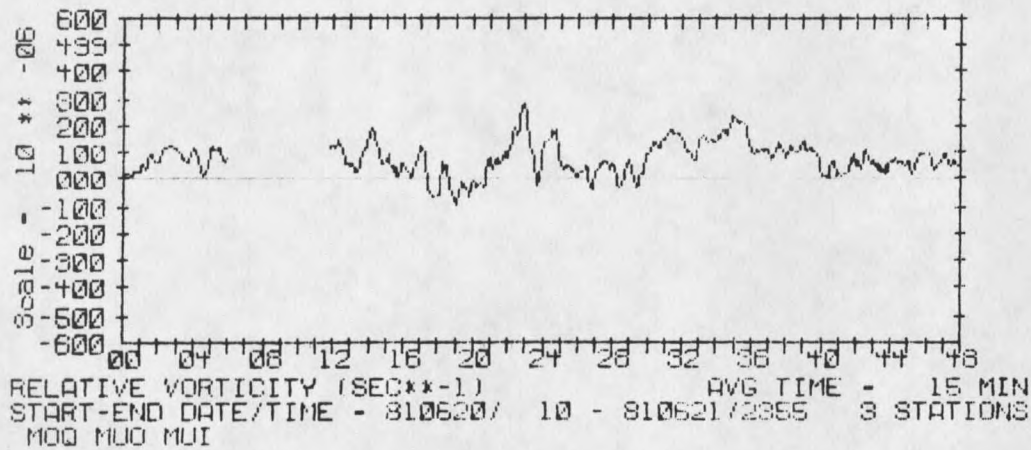
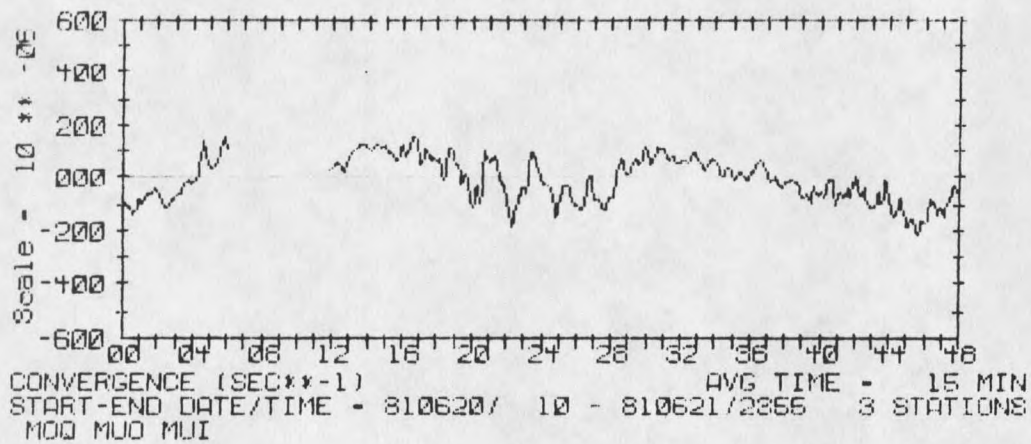
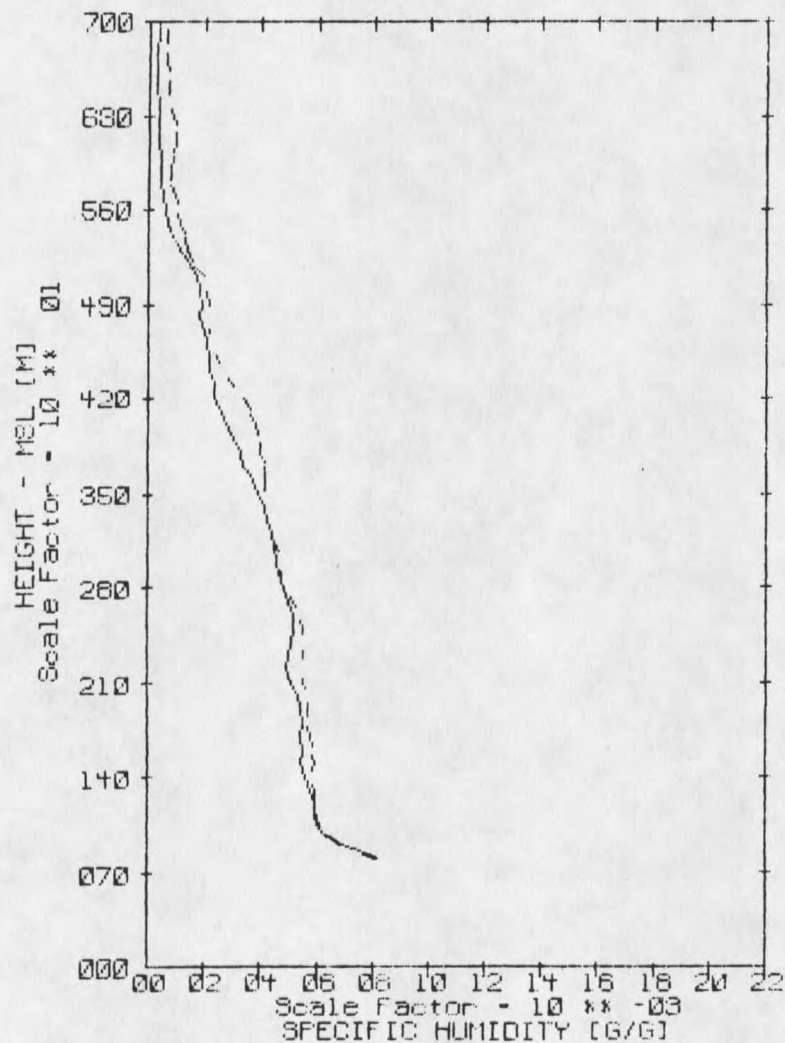


Figure 26. Kinematic convergence, moisture convergence, and relative vorticity for the southwest surface quadrant bounded by MOQ, MUO, and MUI on 20-21 June 1981.

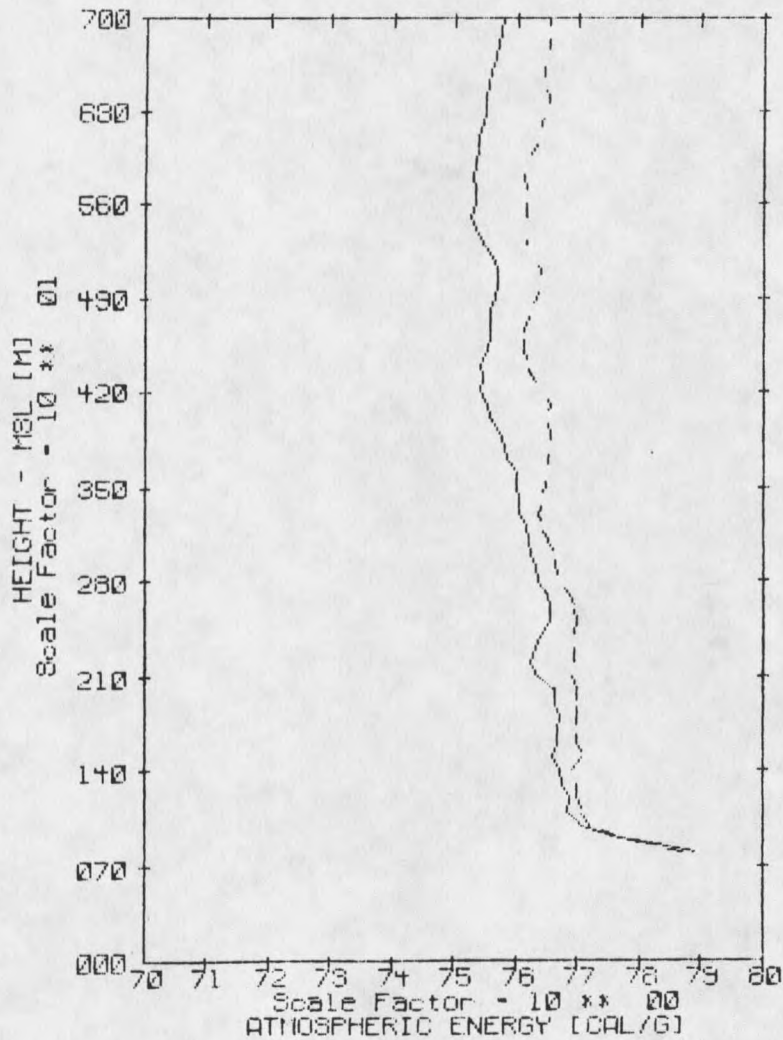
SH vs HM 0 81062021 30/ 30 GMT BAK MLS POW



SH vs HM 1 81062023 44/ 44 GMT BAK MLS POW

Figure 27. Specific humidity at 2130 GMT (solid line) and 2344 GMT (dashed line) on 20 June 1981 for the southern half of the rawinsonde network.

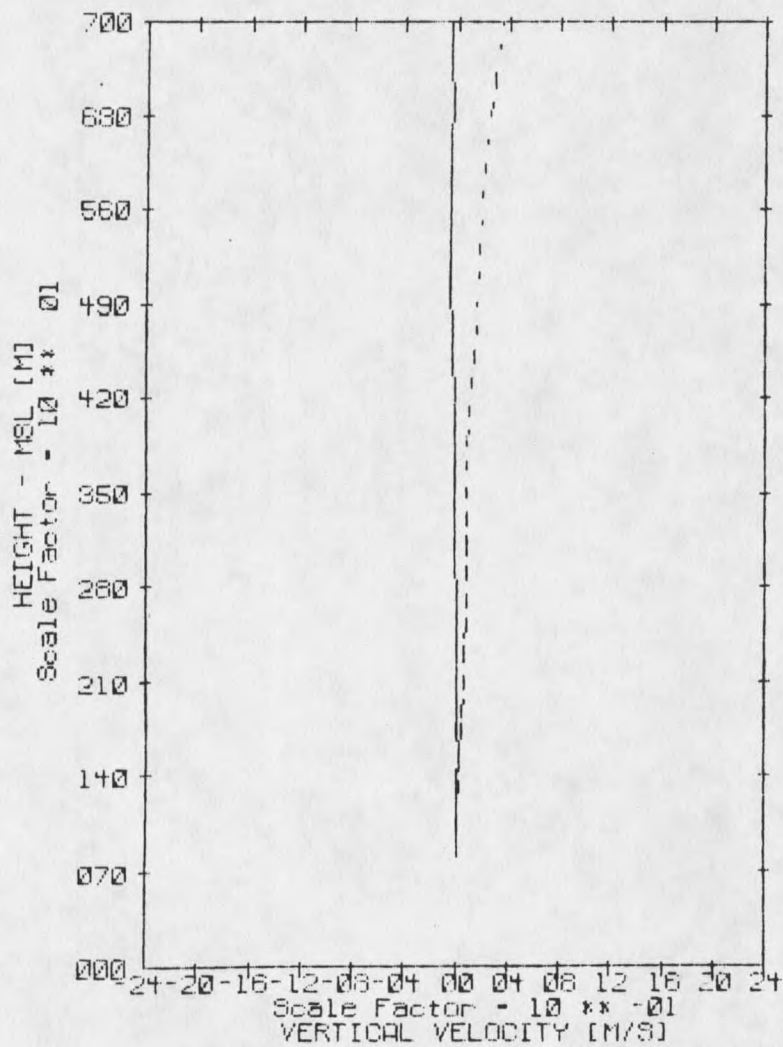
AE vs HM 0 81062021 30/ 30 GMT BAK MLS POW



AE vs HM 1 81062023 44/ 44 GMT BAK MLS POW

Figure 28. Total specific atmospheric energy at 2130 GMT (solid line) and 2344 GMT (dashed line) on 20 June 1981 for the southern half of the rawinsonde network.

WN vs HM 0 81062021 30/ 30 GMT BAK MLS POW



WN vs HM 1 81062023 44/ 44 GMT BAK MLS POW

Figure 29. Vertical velocity at 2130 GMT (solid line) and 2344 GMT (dashed line) on 20 June 1981 for the southern half of the rawinsonde network.

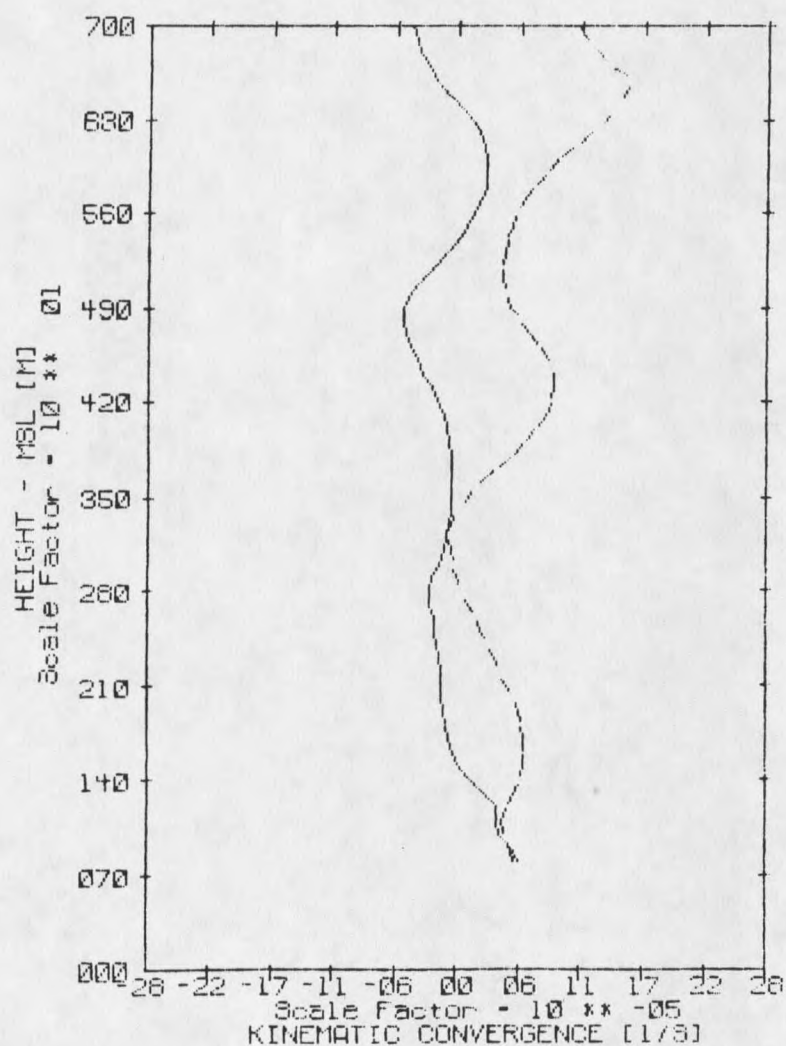
The kinematic convergence (Fig. 30) was relatively formless at 2130 GMT below 3.3 km, but sinusoidal above this level. By 2344 GMT, however, the pattern had become more uniformly sinusoidal throughout, though the wave amplitude remained lower than that occurring during the severe-storm case. The positive vertical velocity is explained by the persistently positive convergence values above 3.4 km. The moisture convergence (Fig. 31) shows the pattern even more clearly. The waves aloft appear to propagate downward, as was seen on the 12th, but the lack of a clear wave pattern below 3.3 km at 2130 makes it difficult to establish conclusive evidence of downward propagating convergence in this layer.

The relative vorticity follows a similar pattern (Fig. 32). The featureless profile at 2130 GMT is replaced by a more sinusoidal pattern at 2344 GMT.

Surface/Upper-air Coupling During the HIPLEX-I Case

The lack of a well-defined sinusoidal pattern in the kinematic convergence trace at 2130 GMT makes it difficult to establish a clear pattern of vertically propagating waves, as was evidenced on the 12th. However, the moisture convergence may indicate that the vertical waves are propagating downward, suggesting that surface/upper-air coupling is becoming established at about the time convective activity was in the area. The increased organization of the upper waves is probably due to the small convective complex that passed through the area during the period. The relative vorticity profile also shows an increasing sinusoidal pattern toward the end of the period. The $1/2$ wavelength lag between the convergence and relative vorticity that was observed during the severe-storm period is also evident at 2344 GMT above 3 km.

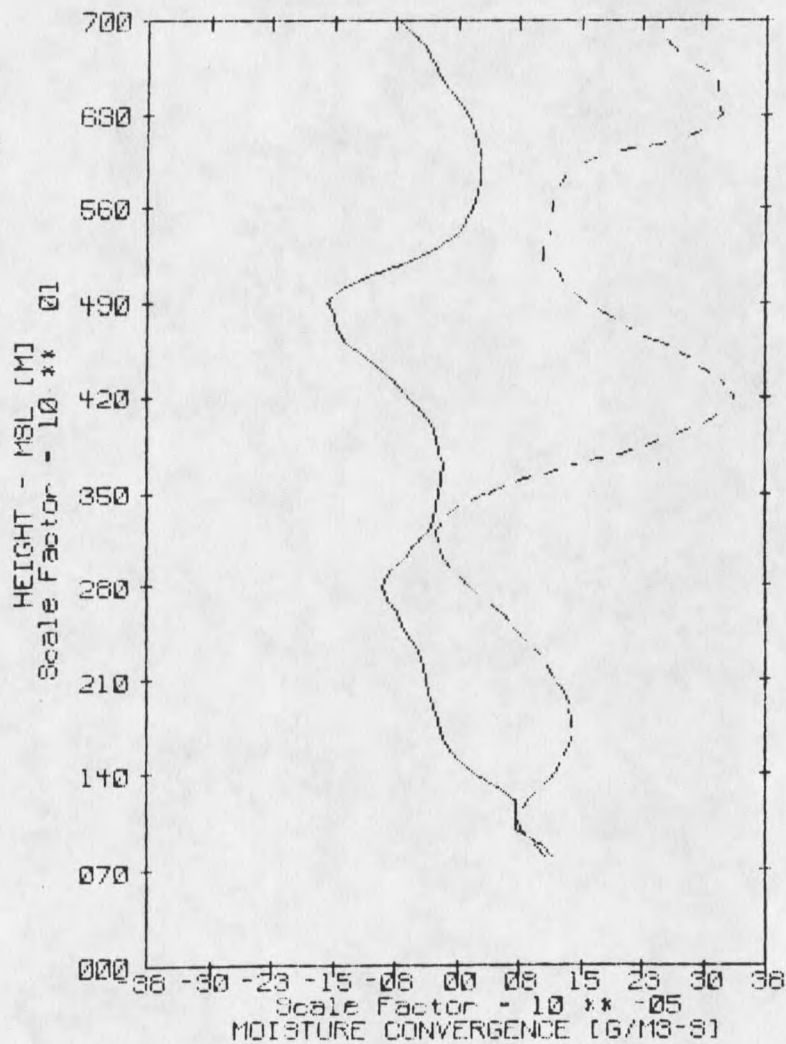
CV vs HM 0 81062021 30/ 30 GMT BAK MLS POW



CV vs HM 1 81062023 44/ 44 GMT BAK MLS POW

Figure 30. Kinematic convergence at 2130 GMT (solid line) and 2344 GMT (dashed line) on 20 June 1981 for the southern half of the rawinsonde network.

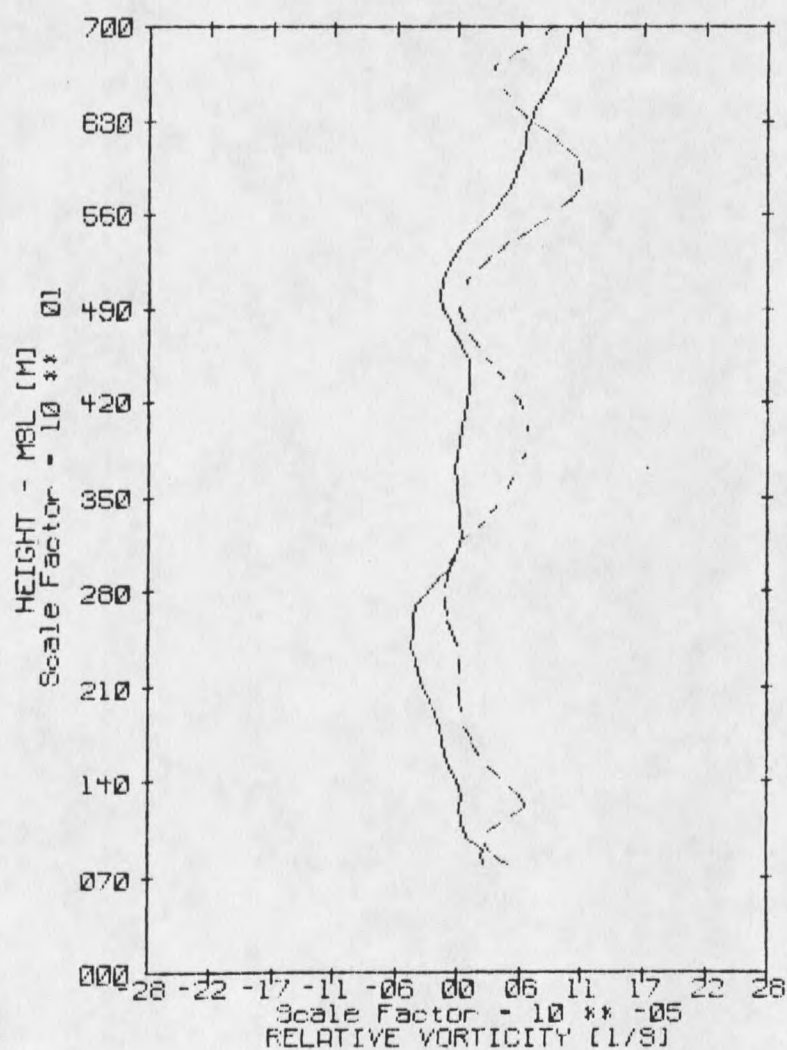
MC vs HM 0 81062021 30/ 30 GMT BAK MLS POW



MC vs HM 1 81062023 44/ 44 GMT BAK MLS POW

Figure 31. Moisture convergence at 2130 GMT (solid line) and 2344 GMT (dashed line) on 20 June 1981 for the southern half of the rawinsonde network.

RV vs HM 0 81062021 30/ 30 GMT BAK MLS POW



RV vs HM 1 81062023 44/ 44 GMT BAK MLS POW

Figure 32. Relative vorticity at 2130 GMT (solid line) and 2344 GMT (dashed line) on 20 June 1981 for the southern half of the rawinsonde network.

Case-Study : The Benign Period (30-31 July 1981)

General Meteorological Conditions During the Benign Case

The morning synoptic analysis for July 30 (Fig. 33) showed a surface low pressure system and associated cold front centered near Miles City. This system was supported by a weak 500 mb trough line slicing through the Idaho panhandle and extending to northern California. The upper low, centered in Alberta, drifted eastward as the surface frontal system moved north-northeastward into the Dakotas. As a result, higher pressure entered the region and the CCOPE mesonet area was left with clear skies and no visible convection throughout the afternoon.

Surface Patterns During the Benign Case

The specific humidity (Fig. 34) peaked early in the morning (1000) in response to the passing frontal system, but steadily dropped as the system progressed eastward. The relative humidity (Fig. 34) showed a diurnal trend similar to that observed during the HIPLEX-I period, but the afternoon peak value was only 50%, as compared to the 80% value on the 20th. The total atmospheric energy (Fig. 34), had early morning values in excess of those recorded at the same time for both of the other two case-study periods (due to the presence of the surface front), but steadily dropped throughout the remainder of the day.

The kinematic convergence (Fig. 35) was mostly negative from 1300 onward, with only a few periods possessing small, positive values. However, the pattern remained sinusoidal. Interestingly, it is not the sinusoidal pattern that distinguishes the benign period from the other cases, but the presence of divergence earlier in the period. The relative vorticity (Fig. 35) was mostly

THURSDAY, JULY 30, 1981

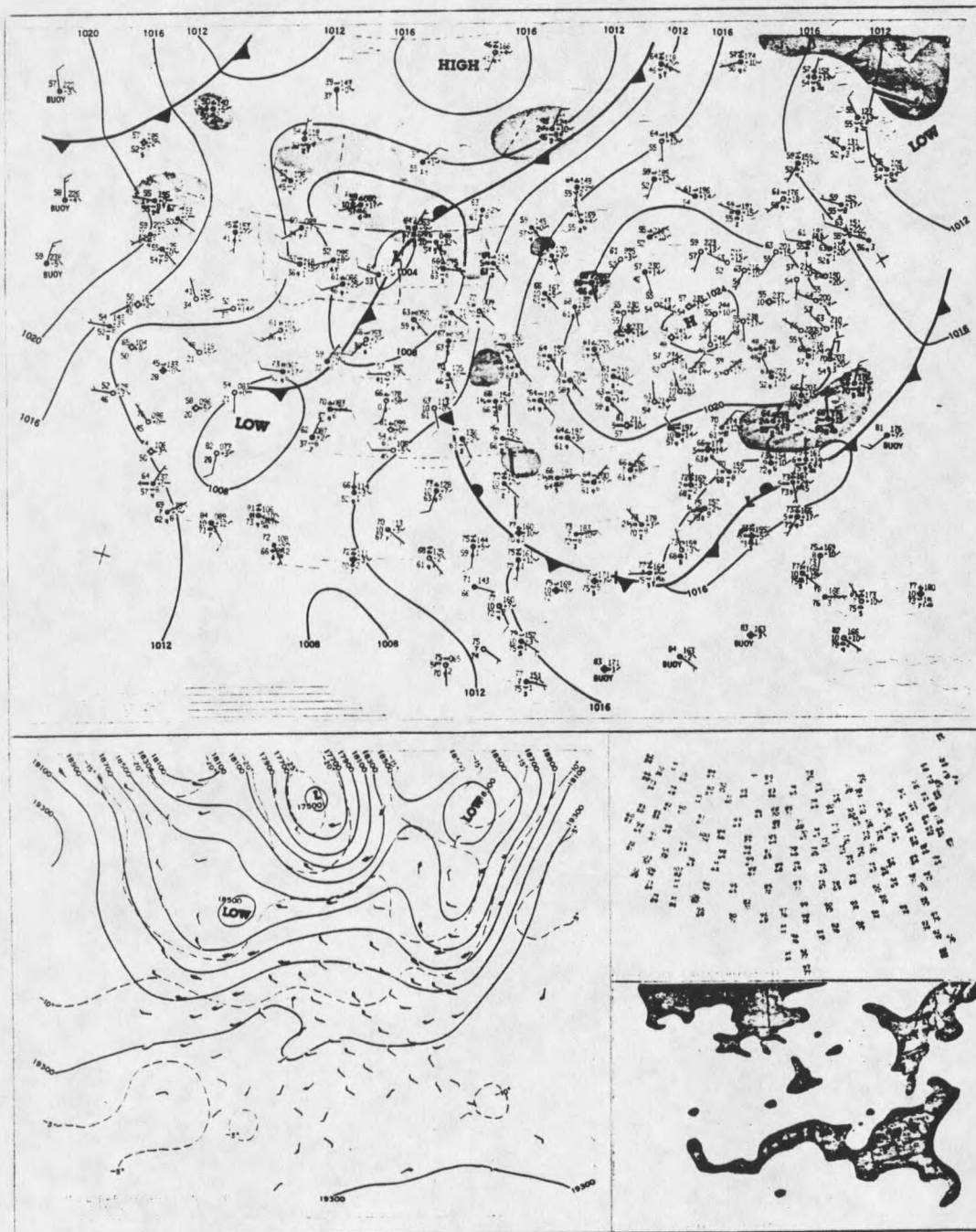


Figure 33. Surface and 500 mb synoptic maps at 1200 GMT on 30 July 1981, with 24 hour temperature extremes and precipitation amounts and areas (U.S. Dept. of Commerce, 1981).

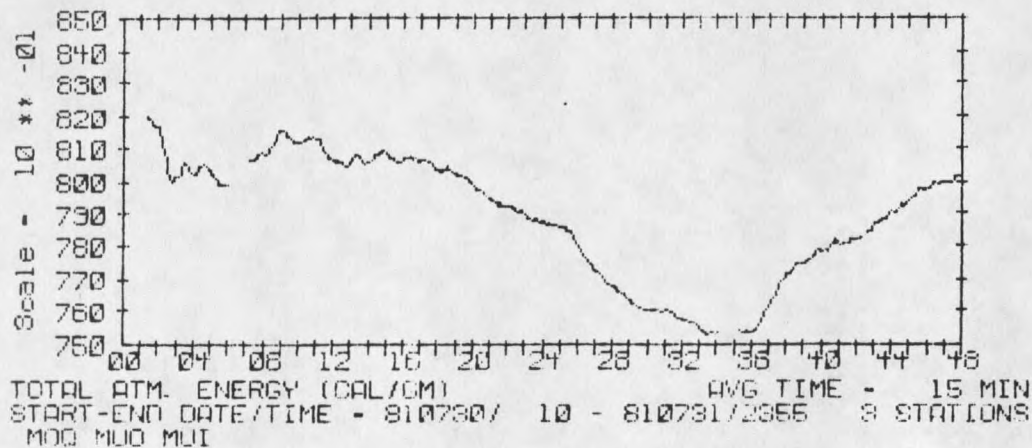
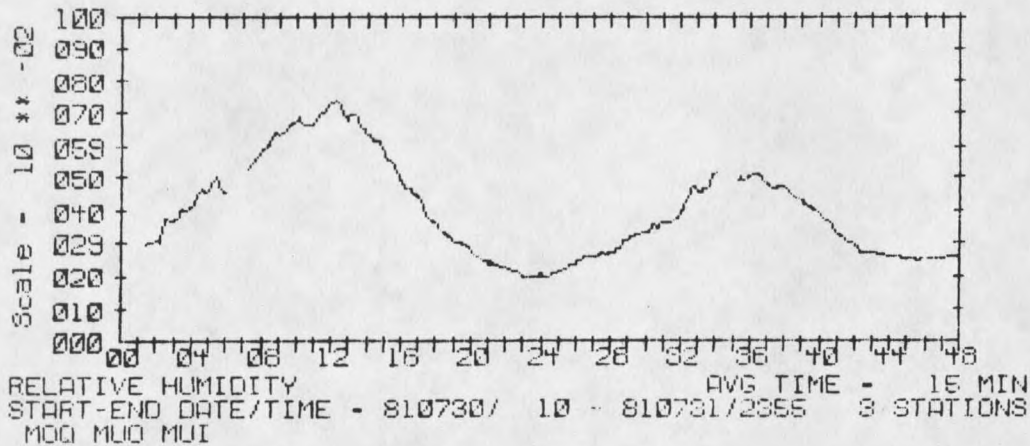
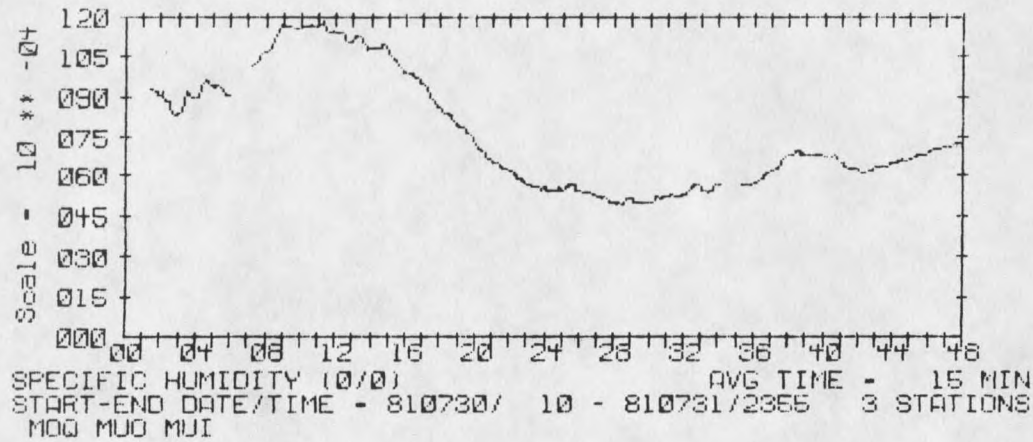


Figure 34. Specific humidity, relative humidity, and total specific atmospheric energy for the southwest surface quadrant bounded by MOQ, MUO, and MUI on 30-31 July 1981.

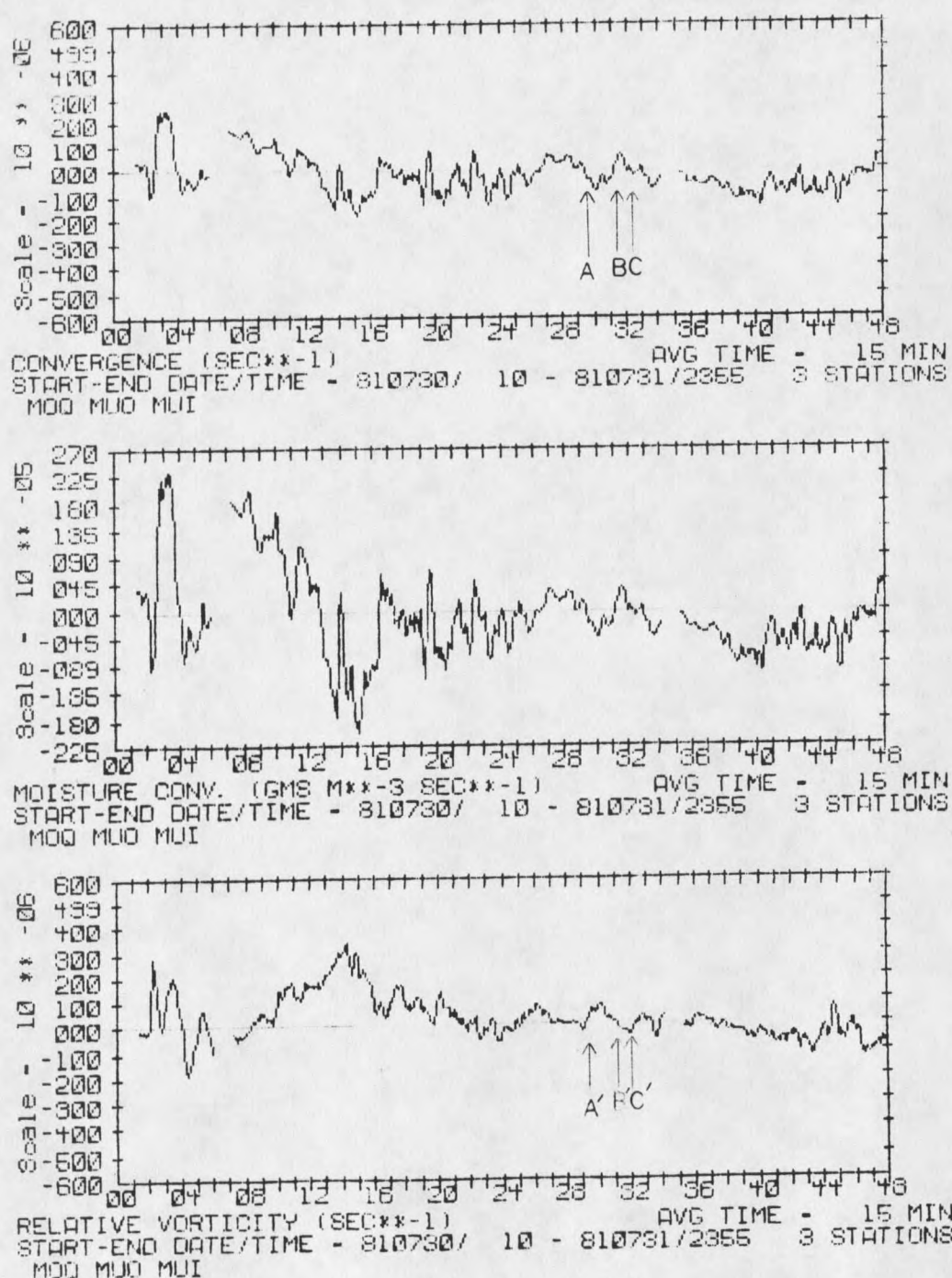


Figure 35. Kinematic convergence, moisture convergence, and relative vorticity for the southwest surface quadrant bounded by MOQ, MUO, and MUI on 30-31 July 1981.

positive during the same period, but the values were at the noise level. The moisture convergence trends (Fig. 35) are almost indistinguishable from the kinematic convergence trends after 2400 because of the low quantity of surface moisture.

Upper-Air Profiles During the Benign Case

Rawinsondes were taken at approximately 90 min intervals from 1830 to 0300 GMT, covering a period three hours longer than for the other cases.

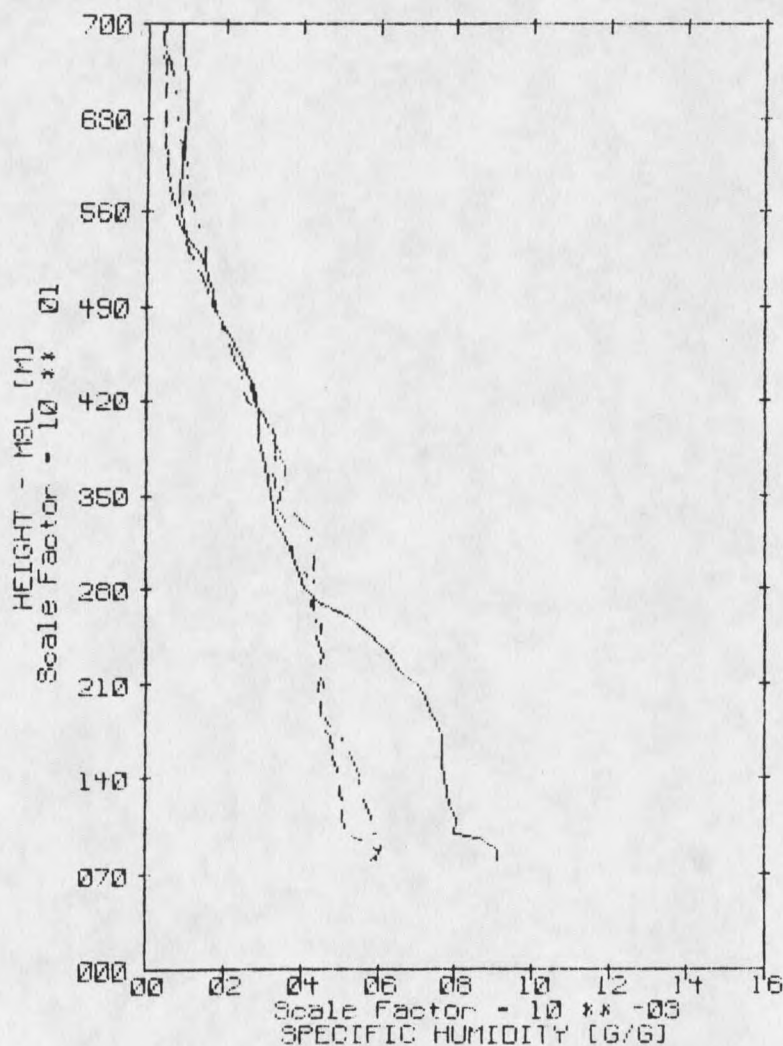
In contrast to the previous two periods, the benign period shows a decrease in the surface water vapor content below 3 km from 1830 to 2340 GMT (Fig. 36). A decrease in atmospheric energy (Fig. 37) occurred during the same period. The profile became even more stable, except for the layer from 4.2 km to 5.5 km, which remained slightly unstable. After 2340 GMT, the profiles remained essentially unchanged.

The kinematic convergence (Fig. 38) and the moisture convergence (Fig. 39) both show well-defined sinusoidal patterns, as does the relative vorticity (Fig. 40). However, unlike the convectively active cases, these waves do not appear to propagate downward at all.

Surface/Upper-Air Coupling During the Benign Case

The sinusoidal kinematic convergence profiles evident in Fig. 38-40, although organized, do not show any evidence of the vertical propagation observed during the previous two periods studied. Values in the lower 1.5 km, however, do undulate between positive and negative values. Because these vertical waves in the upper atmosphere remain stationary, no evidence of coupling is attributed to this period.

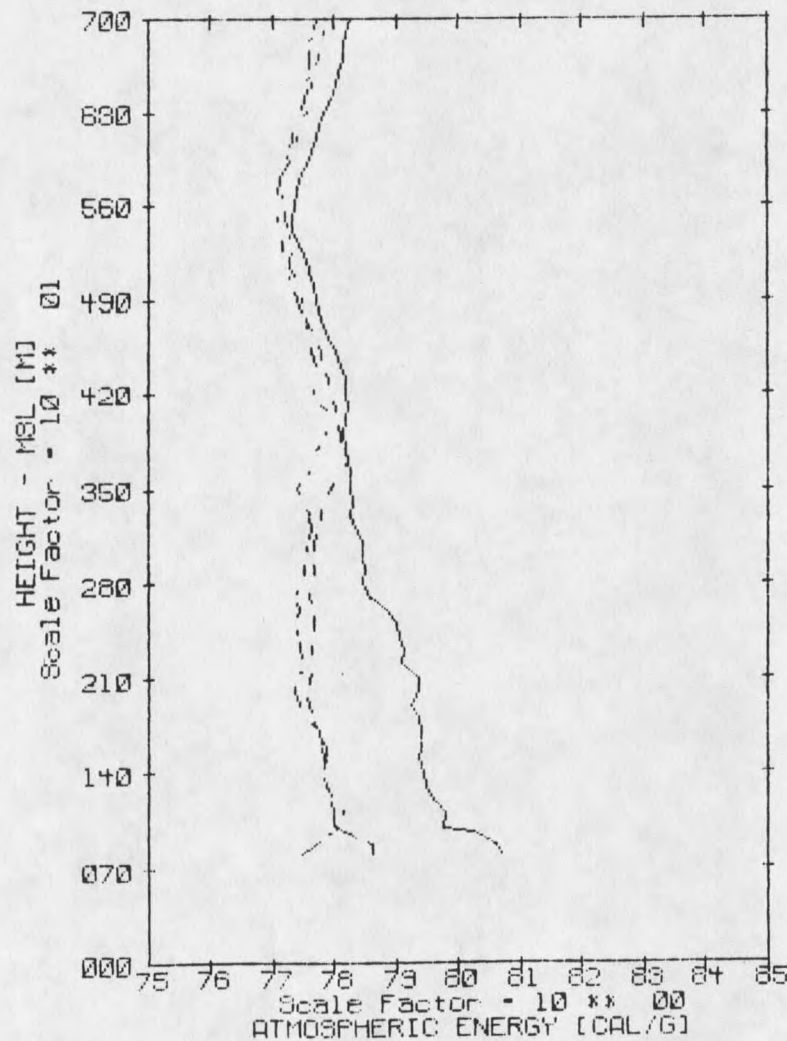
SH vs HM 0 81073018 30/ 30 GMT KWL MLS POW



SH vs HM 1 81073023 40/ 40 GMT KWL MLS POW
SH vs HM 2 81073102 40/ 40 GMT KWL MLS POW

Figure 36. Specific humidity at 1830 GMT (solid line), 2340 GMT (long dashed line), and 0240 GMT (short dashed line) on 30-31 July 1981 for the southwest quadrant of the rawinsonde network.

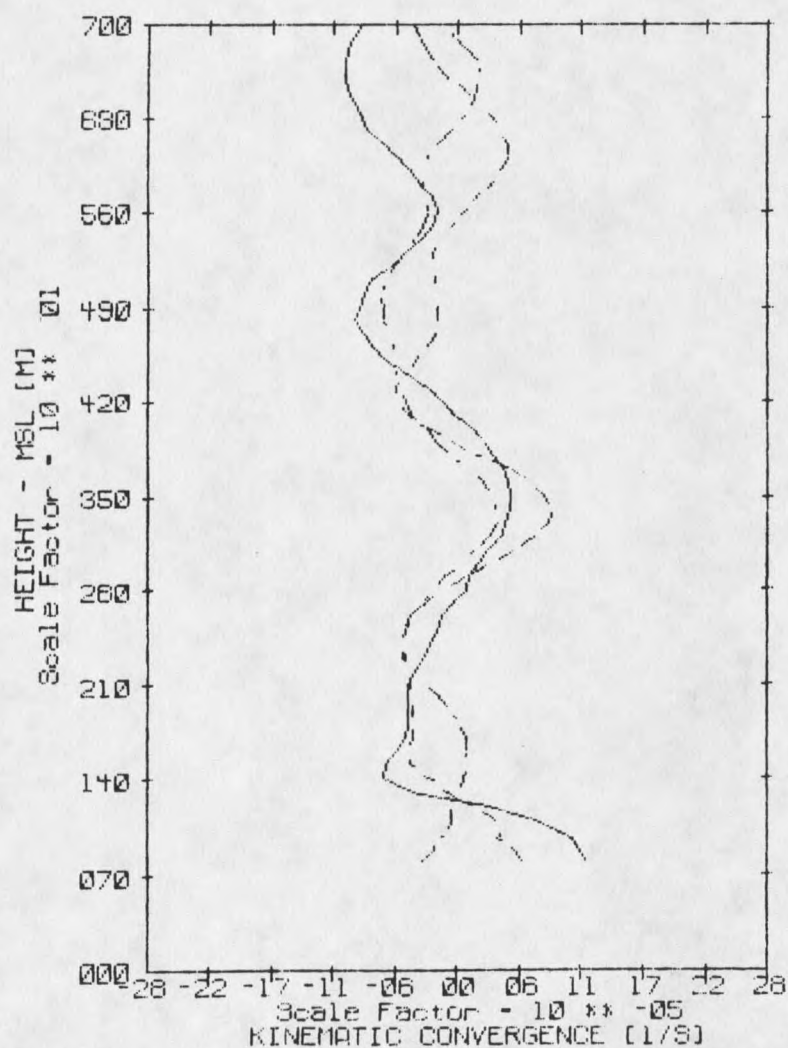
AE vs HM 0 81073018 30/ 30 GMT KWL MLS POW



AE vs HM 1 81073023 40/ 40 GMT KWL MLS POW
AE vs HM 2 81073102 40/ 40 GMT KWL MLS POW

Figure 37. Total specific atmospheric energy at 1830 GMT (solid line), 2340 GMT (long dashed line), and 0240 GMT (short dashed line) on 30-31 July 1981 for the southwest quadrant of the rawinsonde network.

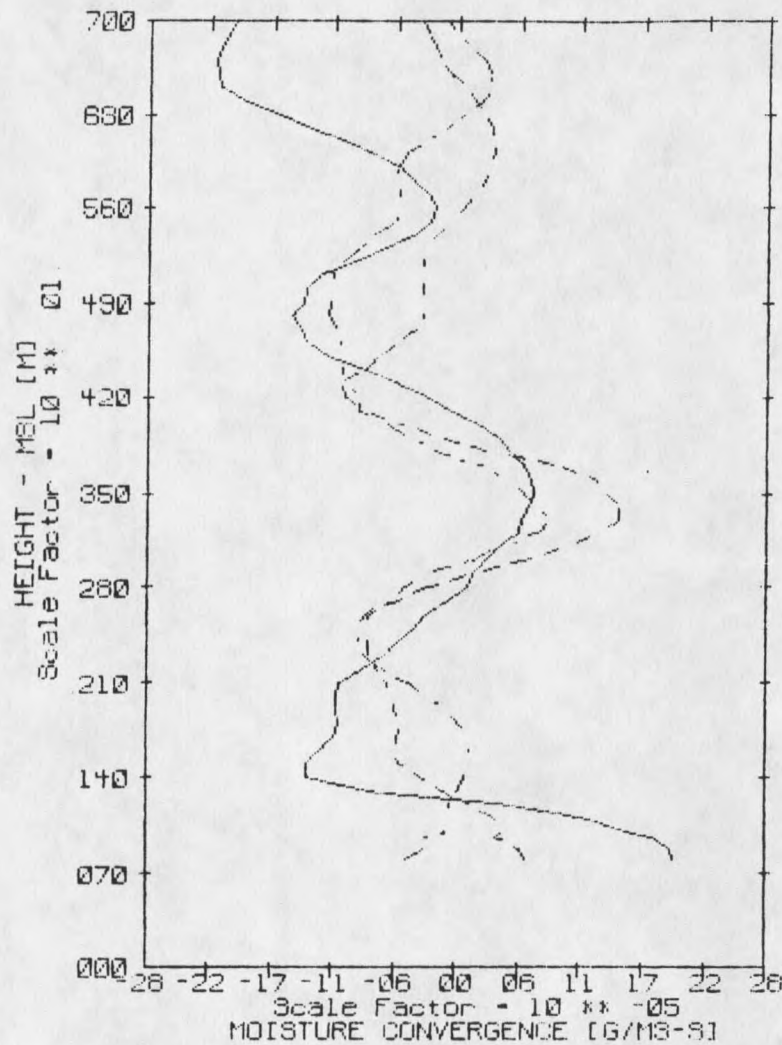
CV vs HM 0 81073018 30/ 30 GMT KWL MLS POW



CV vs HM 1 81073023 40/ 40 GMT KWL MLS POW
CV vs HM 2 81073102 40/ 40 GMT KWL MLS POW

Figure 38. Kinematic convergence at 1830 GMT (solid line), 2340 GMT (long dashed line), and 0240 GMT (short dashed line) on 30-31 July 1981 for the southwest quadrant of the rawinsonde network.

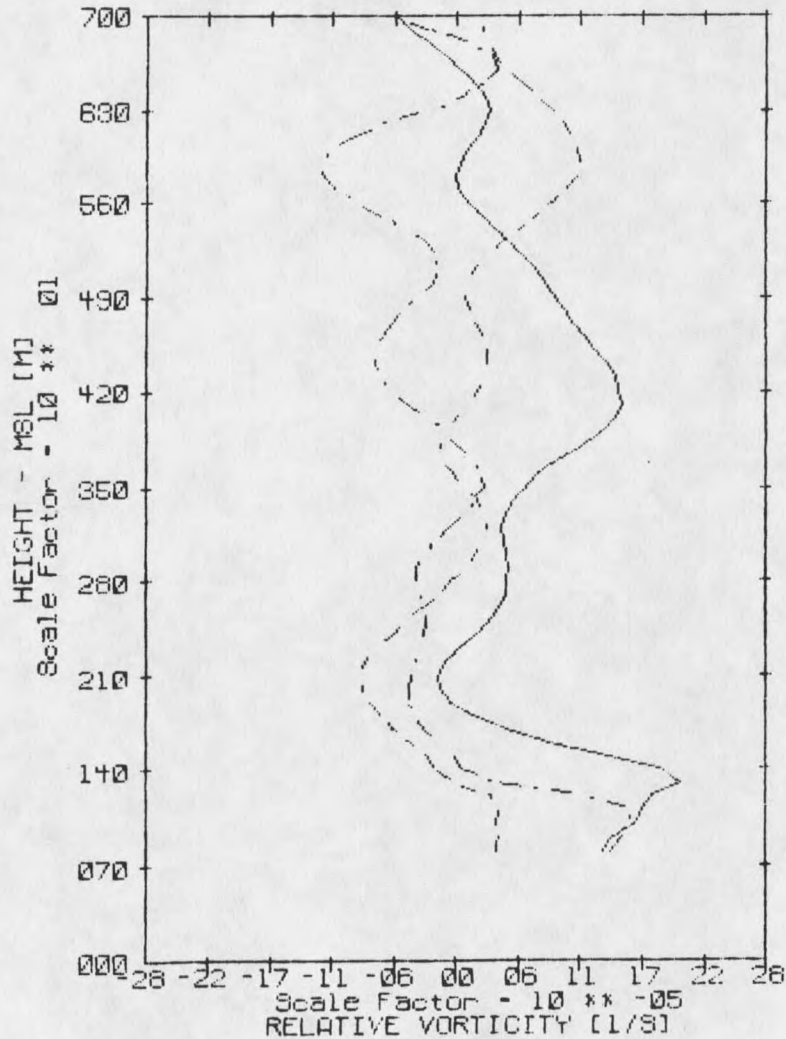
MC vs HM 0 81073018 30/ 30 GMT KWL ML3 POW



MC vs HM 1 81073023 40/ 40 GMT KWL ML3 POW
MC vs HM 2 81073102 40/ 40 GMT KWL ML3 POW

Figure 39. Moisture convergence at 1830 GMT (solid line), 2340 GMT (long dashed line), and 0240 GMT (short dashed line) on 30-31 July 1981 for the southwest quadrant of the rawinsonde network.

RV vs HM 0 81073018 30/ 30 GMT KWL ML3 POW



RV vs HM 1 81073023 40/ 40 GMT KWL ML3 POW
RV vs HM 2 81073102 40/ 40 GMT KWL ML3 POW

Figure 40. Relative vorticity at 1830 GMT (solid line), 2340 GMT (long dashed line), and 0240 GMT (short dashed line) on 30-31 July 1981 for the southwest quadrant of the rawinsonde network.

CHAPTER 6

SUMMARY AND CONCLUSIONS

The 1981 CCOPE surface mesonet and rawinsonde network were compared during three case-study periods to find evidence of surface/upper-air coupling in southeastern Montana. The case-studies covered periods of varying convective activity : A "severe-storm" period (12-13 June) possessing large amounts of precipitation with probable hail; a "HIPLEX-I" period (20-21 June) characterized by the occurrence of the small, semi-isolated convective clouds selected as seeding targets during the HIPLEX-I weather modification experiment; and a "benign" period (30-31 July) with no visible convection.

The method selected to analyze these data was the limited station analysis (LSA) technique, which provided areal averages of primary and derived meteorological parameters. The LSA used a modification of the line integral technique similar to the Bellamy (1949) method to determine the kinematic parameters. Vertical profiles of basic and derived meteorological parameters were generated to search for surface/upper-air coupling.

The LSA technique produced similar characteristics when the bounded area was doubled, and when the number of stations on the area boundary was increased; however, the resultant vertical profiles were somewhat smoothed. Although the number of these comparisons was limited, the results are in agreement with a previous study using surface stations only (Heimbach and Engel, 1986).

The severe-storm period displayed several features observed during other

investigations; most notably a low-level jet, a significant wind-shear layer aloft, advection of a significant quantity of surface moisture, a well-mixed surface boundary layer as evidenced by constant water vapor and equivalent potential energy profiles, an unstable total specific atmospheric energy profile aloft, and a surface temporal pattern of kinematic convergence, relative vorticity, and moisture convergences that is sinusoidal in form. The pre-storm environment contained an alternating pattern of low-magnitude surface convergence/divergence values, with weak anticyclonic vorticity. However, 60 min prior the initiation of convective rainfall, surface convergence was established which increased in intensity until the onset of precipitation. During the convective events, convergence was coupled with cyclonic vorticity, and divergence with anticyclonic vorticity. This agrees with observations made by Ulanski and Garstang (1978).

The upper-air profiles also showed an alternating convergence/divergence regime (see e.g., Byers and Braham, 1949). During the pre-storm environment, no clear coupling between the relative vorticity and kinematic convergence was observed, but just prior to the first convective event, vorticity/convergence coupling became evident, but with a lag of about $1/2$ wavelength. This resembles the "checkerboard" pattern observed by Heimbach and Engel (1986).

A unique finding of this study is the clear evidence of downward propagating sinusoidal convergence and vorticity patterns during the convectively active periods. The waves maintained their identity during their downward movement, but the absolute values of the relative maxima and minima were not conserved. Rather, their magnitude changed as they lowered and eventually interacted with the surface layer. A direct point-by-point comparison of the temporal surface plots with the upper-air values produced

limited agreement because of the different averaging times (instantaneous versus 15 min), siting considerations, and boundary layer interactions. However, the patterns of the upper-air features showed good agreement with the corresponding surface patterns. The continuous propagation of the vertical waves through the surface mixing layer suggests that these events were coupled to the surface. A downward flux of moisture, evident in the moisture convergence profile, supports this conclusion.

The HIPLEX-I period lacked the low-level jet and wind shear noted during the severe-storm period. In addition, less moisture advection and a lower atmospheric energy were apparent. The surface convergence remained mostly positive before the small convective complex entered the region, but changed to an alternating divergence/convergence pattern during the event. After convection had been established, the surface vorticity/convergence coupling observed during the severe-storm period, as well as the $1/2$ wavelength lag in the vertical profiles were evident.

The downward movement of the vertical sinusoidal waveforms in the kinematically-derived profiles was also observed during the HIPLEX-I period. However, the pattern was not as clearly established in the lower levels of the profiles until the end of the sampling period.

The benign period was characterized by lowering atmospheric energy and specific humidity profiles during the afternoon, in contrast to the other cases. The upper-air profiles of the kinematically-derived profiles also showed a well-organized sinusoidal pattern; however, the waves above the surface layer showed little vertical propagation throughout the sampling period.

Comparison of the vertical profiles of kinematic convergence, relative vorticity, and moisture convergence during the three case-study periods analyzed in this study suggest that there is surface/upper-air coupling during convectively active periods. The evidence strengthens with the degree of convection. Little or no indication of coupling is apparent during convectively benign periods. The surface/upper-air coupling is manifest as groundward propagating sinusoidal convergence and vorticity patterns passing through the surface layer, and is observed as a sinusoidal temporal pattern on the surface.

The increasing convergence and vorticity values in Fig. 9 between AB, and A'B'; the corresponding negative peaks at B and B'; and the change from positive to negative values at points C and C' indicate coupling between these two parameters during convectively active periods. Fig. 35, however, shows negative convergence values between AB and mostly positive vorticity values between corresponding points A'B', with the reverse between BC and B'C', suggesting that surface convergence and vorticity are not coupled during convectively benign periods. The best evidence for surface/upper-air coupling based on surface information alone, then, appears to be the extent of convergence/vorticity coupling evident on the temporal surface traces.

REFERENCES CITED

REFERENCES CITED

- Acherman, B., G.L. Actemeier, H. Appleman, S.A. Changonon, F.A. Huff, G.M. Morgan, P.T. Scheckdanz, and R.G. Semonin, 1976: Design of the High Plains Experiment with Specific Focus on Phase 2, Single Cloud Experimentation. Final Report on HIPLEX Design Project, prepared for the Division of Atmospheric Water Resources Management, Bureau of Reclamation, Denver, CO, June 30, 1976, 231 pp.
- Achtemeier, G.L., 1980: Boundary Layer Structure and its Relation to Precipitation Over the St. Louis Area. Technical Report 2, State Water Survey Division, Illinois Institute of Natural Resources, Urbana, Ill., 61801, 275 pp.
- Bellamy, J.C., 1948: Objective calculations of divergence, vertical velocity, and vorticity. Bull. Amer. Meteor. Soc., 5, 304-307.
- Betts, A.K., 1973: Non-precipitating cumulus convection and its parameterization. Quart. J. Roy. Meteor. Soc., 99, 178-196.
- Byers, J. and R.R. Braham, 1949: The Thunderstorm. U.S. Government Printing Office, Washington, D.C., LC 50-60624, 287 pp.
- Cooper, H.J., M. Garstang, and J. Simpson, 1982: The Diurnal interaction between convection and peninsular-scale forcing over South Florida. Mon. Wea. Rev., 110, 486-503.
- Darkow, G.L., 1968: The total energy environment of severe storms. J. Appl. Meteor., 7, 199-205.
- Dennis, A.S., E.W. Holroyd, III, W.E. Howell, D.A. Matthews, B.A. Silverman, and A.B. Super, 1984: HIPLEX: A Cooperative Program on Rain Augmentation in the High Plains (Final Report), Bureau of Reclamation, Division of Atmospheric Resources Research, Denver Federal Center, Engineering and Research Center, Denver, CO, 55 pp.
- Doneaud, A.A., J.R. Miller, D.L. Priegnitz and L. Viswanath, 1983: Surface mesoscale features as potential storm predictors in the Northern Great Plains - Two case studies. Mon. Wea. Rev., 111, 273-292.
- Fankhauser, J.C., 1969: Convective processes resolved by a mesoscale rawinsonde network. J. Appl. Meteor., 8, 778-798.
- _____, 1982: Hail Storms of the Central High Plains, 2, 1-34. Available from the National Center for Atmospheric Research, Boulder, CO.

REFERENCES CITED Continued

- Fuelberg, H.E., and M.F. Printy, 1983: Meso β -scale thunderstorm/environment interactions during AVE-Sesame V (20-21 May 1979), Bull. Amer. Meteor. Soc., 64, 1144-1156.
- Hartzel, C.L., 1981: Meteorological Services Supporting Weather Modification Research at Miles City, MT Progress Report for Feb '81, Western Weather Consultants, Inc., Durango, CO 81301.
- Heimbach, J.A., Jr., and A.B. Super, 1980: Rain gage network requirements from a simulated convective complex experiment, J. Appl. Meteor., 19, 1170-1183.
- Heimbach, J.A., Jr., and T. Engel, 1981: Application of CCOPE 1981 Mesonet Data to Describe Physical Processes of Convective Storm Initiation and Evolution., Proposal to the National Science Foundation, Experimental Meteorology Branch, 43 pp.
- _____, 1986: The use of limited surface networks to measure mesoscale phenomenon, submitted to Mon. Wea. Rev..
- Hess, S.L., 1959: Introduction to Theoretical Meteorology, Holt, Rhinehart and Winston, 362 pp.
- Holle, R.L. and A.I. Watson, 1983: Duration of convective events related to visible cloud convergence, radar and rain gage parameters in south Florida, Mon. Wea. Rev., 111, 1046-1051.
- Holton, J.R., 1972: An Introduction to Dynamic Meteorology, Academic Press, New York, 319pp.
- House, D.C., 1960: Remarks on the optimum spacing of upper-air observations, Mon. Wea. Rev., 88, 97-100.
- Mahrt, L., and D. Pierce, 1980: Relationship of moist convection to boundary-layer properties: application to a semiarid region, Mon. Wea. Rev., 108, 1810-1815.
- Matthews, D.A., 1983: Analysis and Classification of Mesoscale Cloud and Precipitation Systems, Bureau of Reclamation, Engineering and Research Center, P.O. Box 25007, Denver, CO 80225, 422 pp.
- Milner, S., 1986: NEXRAD, the coming revolution in radar storm detection and warning, Weatherwise, 39, 72-85.
- National Academy of Sciences, Committee on Atmospheric Sciences and Assembly of Mathematics and Physical Sciences, 1980: The Atmospheric Sciences: National Objectives for the 1980's, 9-72, Washington, D.C., 130 pp.

REFERENCES CITED Continued

- Newton, C.W., 1963: Dynamics of severe convective storms, Meteor. Monogr., 5, No. 27, 33-64.
- O'Brien, J.J., 1970: Alternative solutions to the classical vertical velocity problem, J. Appl. Meteor., 9, 197-203.
- Orlanski, I., 1975: A rational subdivision of scales for atmospheric processes, Bull. Amer. Meteor. Soc., 56, 527-530.
- Retallack, B.J., 1971: Compendium of Lecture Notes for Training Class III Meteorological Personnel, WMO-NO. 291, 381 pp.
- Saucier, W.J., 1972: Principals of Meteorological Analysis, University of Chicago Press, Chicago, Ill. 60637, 438 pp.
- Schaefer, J.T., 1973: On the computation of the surface divergence field, J. Appl. Meteor., 12, 546-547.
- Scoggins, J.R., J.E. Fuelburg, S.F. Williamson, and M.E. Humbert, 1978: Mesoscale Characteristics of the Texas Hiplex Area During Summer 1976, Final Report, LP-65, prepared for the Bureau of Reclamation, Texas Department of Water Resources, Austin, TX p. 322-343.
- Smith, P.L., A.S. Dennis, B.A. Silverman, A.B. Super, E.W. Holroyd, III, W.A. Cooper, P.W. Mielke, Jr., K.J. Berry, H.D. Orville, and J.R. Miller, Jr., 1984: HIPLEX-I: Experimental Design and Response Variables. J. Climate Appl. Meteor., 23, 497-512.
- Tabata, S., 1973: A simple but accurate formula for the saturation vapor pressure over liquid water, J. Appl. Meteor., 12, 1410-1411.
- Ulanski, S.L., and M. Garstang, 1978: The role of surface divergence and vorticity in the life cycle of convective rainfall. Part I: Observation and analysis. J. Atmos. Sci., 35, 1047-1062.
- U.S. Dept. of Commerce; NOAA/EDIS, 1981: Daily Weather Maps, Weekly Series, June 8-14, 1981; June 15-21, 1981; July 27-August 2, 1981; U.S. Govt. Printing Office, Washington, D.C. 20402.
- U.S. Dept. of Int., Bureau of Reclamation, 1982: 1981 CCOPE Data Inventory, Cooperative Convective Precipitation Experiment, P.O. Box 25007, DFC, Denver, CO 80225, 451 pp.
- Watson, A.I., and D.O. Blanchard, 1984: The relation between total area divergence and convective precipitation in south Florida. Mon. Wea. Rev., 112, 673-685.

REFERENCES CITED Continued

Woodley, W.L., and R.I. Sax, 1976: The Florida Area Cumulus Experiment: rationale, design, procedures, results, and future course. NOAA Tech. Rep. ERL 354, WMPO-6, 204 pp.

APPENDICES

APPENDIX A

RAWINSONDE PROGRAM LISTINGS

APPENDIX A

RAWINSONDE PROGRAM LISTINGS

Rawinsonde mainline programsRAWIN task building executive

```

C*****PROGRAM RAWINT
*****
* RAWIN task building executive                        840820 Green *
CALLS RWTSKZ                                           *
CALLS RWOPCI                                           *
CALLS RWGETL                                           *
CALLS RWFLGI                                           *
CALLS DXNUM                                           *
CALLS DFDRV                                           *
CALLS RWTSKI                                           *
CALLS RWPLTI                                           *
CALLS RWTSKD                                           *
CALLS RWTSKO                                           *
CALLS RWTSKW                                           *
CALLS RWJOBF                                           *
*****
PROGRAM RAWINT

```

```

$INCLUDE: 'RAWINTSK.DCL'

```

```

CHARACTER BELL*1,ANS*1
INTEGER TSKNUM,TSTBIT,DXNUM,CONSLU,STATUS,PONUM

```

```

DATA CONSLU /0/

```

```

90000 FORMAT(A)
90010 FORMAT(A,A,°)
90020 FORMAT(F7.2)
90030 FORMAT(I3)

```

```

*Begin RAWINT

```

```

TSKNUM = 1
WRITE(*,90000) 'ORAWIN (840820) CCOPE81 Rawinsonde Processing'
WRITE(*,90000) ' RAWIN_RAWINT - task building module'

```

```

C Initialization

```

```

CALL RWTSKZ
BELL = CHAR(7)

```

C PO Input processing option

```
10000 TSTBIT = BIT0
      CALL RWOPCI(POLEN,POOPC,PODES,POBIT,POTTL,TSTBIT,PONUM)
      GOTO (10000,15000,20000,25000,30000,35000,40000,45000,
-        50000,55000,60000,65000,80000) PONUM + 1
```

C PO=NP Processing option = new processing period

```
15000 CALL RWGETL(.TRUE.,TTINT)
      GOTO 10000
```

C PO=SF Processing option = set processing flags

```
20000 CALL RWFLGI
      GOTO 10000
```

C PO=SP Processing option = set fetch upper pressure level

```
25000 WRITE(*,90010) '0Input fetch upper pressure level : ',BELL
      READ (*,90020) SNDPRL
      GOTO 10000
```

C PO=SX Processing option = set sounding file drive

```
30000 WRITE(*,90010) '0Input SOUNDING file drive letter or ¼CR½: ',BELL
      READ (*,90000) ANS
      SNDDX = DXNUM(ANS)
      IF (SNDDX .EQ. 0) THEN
        CALL DFDRV(I)
        SNDDX = I
      ENDIF
      GOTO 10000
```

C PO=DM Processing option = set display mode

```
35000 TSTBIT = BIT0
      CALL RWOPCI(DMLen,DMOPC,DMDen,DMBIT,DMTTL,TSTBIT,DMNUM)
      TSNUM = 0
      GOTO 10000
```

C PO=TS Processing option = set task specifications

```
40000 CALL RWTSKI
      DO 40010 I = 1, TBCMAX
        PFCSPC(I) = 0
40010 CONTINUE
      IF ( (DMNUM .EQ. PL) .OR. (DMNUM .EQ. BO) ) THEN
        IF (TSNUM .NE. 0) THEN
          WRITE(*,90000) '0Plot specifications : '
          GOTO 45000
        ENDIF
      ELSE
        GOTO 10000
      ENDIF
```

```

C PO=PF Processing option = set plot format
45000 IF (DMNUM.EQ. TB) THEN
      WRITE(*,90000) '0Plot display mode not enabled'
      GOTO 10000
    ELSEIF (TSNUM.EQ. 0) THEN
      WRITE(*,90000) '0Task specifications not entered'
      GOTO 10000
    ENDIF
    CALL RWPLTI
    GOTO 10000

C PO=DF Processing option = set default task specs
50000 CALL RWTSKD
      GOTO 10000

C PO=DT Processing option = display current task specs
55000 CALL RWTSKO(CONSLU,TSKNUM)
      GOTO 10000

C PO=WT Processing option = write task to task file
60000 CALL RWTSKW(TSKNUM)
      GOTO 10000

C PO=TI Processing option = set sounding time interval
65000 WRITE(*,90010) '0Input time interval (1/40 disables) : ',BELL
      READ (*,90030) TTINT
      IF (TTINT.GT. 55) TTINT = 55
      GOTO 10000

C PO=EN Processing option = end task description
80000 TSKNUM = TSKNUM - 1
      CALL RWJOBFB(TSKNUM,1,1,0)

*Terminate RAWINT
  STOP 'RAWIN_RAWINT -- Normal termination'
  END

```

RAWIN Job processing executive

```

C*****PROGRAM RAWINJ
C * * * * *
C 860504 ODGreen RAWIN job processing executive MS FORTRAN 3.20
CALLS DFDRV
CALLS RWJOBZ
CALLS RWJOBFB
CALLS RWTSKR
CALLS RWTSKO
CALLS RWDMPB
CALLS RWGETA

```

```

CALLS RWMSLC
CALLS RWYYXB
CALLS BITVAL
CALLS RWPMYC
CALLS RWNTRP
CALLS RWPMXC
CALLS RWTBLO
CALLS RWTMPF
CALLS NCOD11
C * * * * *
C
      PROGRAM RAWINJ
C
$INCLUDE: 'RAWINJOB.DCL'
C
      INTEGER TSKNUM, CONSLU, BITVAL, TMPNUM, SCC, TMPLU
      CHARACTER SCCTXT(12)*1, C(11)*1
C
      DATA CONSLU/0/, TMPLU/5/
C
90000 FORMAT(A)
90005 FORMAT('OProcessing fetch list entry ', I10)
90010 FORMAT(1X, 'RAWIN_RAWINJOB : ', A14, 1X, 'Unit', 1X, I4, 1X, A)
90020 FORMAT(1X, 4I10)
C
*Begin RAWINJOB

      CALL DFDRV(TMPDX)
      TMPNUM = 0
      WRITE(*, 90000) 'ORAWIN(840820) CCOPE81 Rawinsonde Processing'
      WRITE(*, 90000) ' RAWIN_RAWINJOB Taskfile processing'

C Initialize common and read the jobfile.
      CALL RWJOBZ
      5000 CALL RWJOBF(NUMTSK, TSKNUM, GETNUM, 1)

C Read in the task description
      10000 CALL RWTSKR(TSKNUM)
      WRITE(*, 90000) 'ORAWIN_RAWINJOB Current task description'
      CALL RWTSKO(CONSLU, TSKNUM)
      WRITE(*, 90005) GETNUM

C Fetch sounding data, calculate MSL heights, and build the master
C ordinate array.
      IF(FDFLG) CALL RWDMP
      CALL RWGETA(STNREQ, PSFLG)
      CALL RWMSLC
      CALL RWYYXB

C Calculate the ordinate column for an IP table

```

```

      IF (TSNUM .EQ. IP) THEN
        IC = 0
        DO 15000 I = 1, STNMAX
          IF (BITVAL(STNREQ,I-1) .EQ. SET) THEN
            IC = IC + BITVAL(GETSTN(GETNUM),I-1)
          ENDIF
15000    CONTINUE
          IF (IC .EQ. 0) THEN
            GOTO 89000
          ELSE
            CALL RWPMYC(STNREQ)
          ENDIF
        ENDIF
      ENDIF

C One table is built for each table header
      DO 18000 TBHNUM = 1, TBHLEN

C Calculate the ordinate column for an MP table
      IF (TSNUM .EQ. MP) THEN
        IC = 1
        DO 15500 I = 1, STNMAX
          IF (BITVAL(TBHSPC(TBHNUM),I-1) .EQ. SET) THEN
            IC = IC * BITVAL(GETSTN(GETNUM),I-1)
          ENDIF
15500    CONTINUE
          IF (IC .EQ. 0) THEN
            GOTO 18000
          ELSE
            CALL RWPMYC(TBHSPC(TBHNUM))
          ENDIF
        ENDIF
      ENDIF

C Process each table line
      DO 17000 TBLLIN = 1, TBLLEN

C Interpolate to find primary data values at current ordinate
      CALL RWNTRP

C Calculate abscissa values for each table column
      DO 16000 TBCNUM = 1, TBCLEN
        IF (TSNUM.EQ.MP) CALL RWPMXC(TBHSPC(TBHNUM),TBCSPC(TBCNUM),V)
        IF (TSNUM.EQ.IP) CALL RWPMXC(TBCSPC(TBCNUM),TBHSPC(TBHNUM),V)
        TBLVAL(TBLLIN,TBCNUM+1) = V
16000    CONTINUE

C Next table line
17000    CONTINUE

C Output the table
      IF ( (DMNUM .EQ. TB) .OR. (DMNUM .EQ. BO) ) THEN

```

```

      CALL RWTBLO(TSKNUM)
    ENDIF

C Output temporary file (plot communication)
    IF ( (DMNUM .EQ. PL) .OR. (DMNUM .EQ. BO) ) THEN
      CALL RWTMPF(TMPNUM)
    ENDIF

C Next table header
18000 CONTINUE

C Process next fetch list entry
    OPEN(TMPLU,FILE='RAWINJOB.DTA',STATUS='OLD')
    WRITE(*,90010) 'RAWINJOB.DTA',TMPLU,'Opened'
    READ(TMPLU,90020) N,I,J,K
    J=J+1
    IF(J .GT. K) THEN
      J=1
      I=I+1
    ENDIF
    REWIND(TMPLU)
    WRITE(TMPLU,90020) N,I,J,K
    CLOSE(TMPLU)
    WRITE(*,90010) 'RAWINJOB.DTA',TMPLU,'Closed'
    IF(J .NE. 1) GOTO 5000

*Terminate RAWINJOB
89000 STOP ' RAWIN_RAWINJOB normal termination'
      END

```

RAWIN file interpolation module

```

C*****PROGRAM RAWINI
C * * * * *
C 860409 ODGreen                      IBM/PC  MS' FORTRAN 3.20 *
C Interpolate between two soundings to get values at a specified time.*
C *
C Source (input) files are those produced by the RAWIN default task *
C specifications. *
C *
C Interpolated data is output to file in RAWIN format. User must *
C construct dummy index file before using these files with the other *
C RAWIN modules. *
C *
C Note: User must specify all 14 characters for filespec *
C *
C Subprogram references : *
CALLS MINYR   in dataset UTLTYLBR.FOR (odg)      /&RAWINI.FOR/ *
CALLS XINTRP  in dataset UTLTYLBR.FOR (odg)      /&RAWINI.FOR/ *

```

```

CALLS UUVV      in dataset METLBR.FOR   (odg)      /&RAWINI.FOR/      *
CALLS LOCAL     in dataset UTLTYLBR.FOR (odg)      /&RAWINI.FOR/      *
CALLS WIND      in dataset PMKINAD.FOR  (jah)      /!RAWINI.FOR/      *
C                                                       *
C Link stream:   RAWIN+&RAWINI+!RAWINI...          *
C * * * * *                                           *
C
      PROGRAM RAWINI
C
      CHARACTER   SRCFL1*14,SRCFL2*14,DSTFIL*14,CHR*1,DTAFMT*20
      REAL        DTA(400,7,3)
      INTEGER     CRDNUM,DTAFLD,MINYR

      DATA LUFIL /5/
C
90000 FORMAT(A)
90010 FORMAT(A,*)
90020 FORMAT(A14)
90030 FORMAT('ORAWINI :',1X,A14,1X,'Unit ',I2,1X,A)
90040 FORMAT(I6)
90050 FORMAT(2X,I6,I2,1X,I2)
90060 FORMAT(13X,A1)
91000 FORMAT(I8,I2,A1,' ',10X,I8,F10.0,2X,I8,20X)
91010 FORMAT(I8,6F5.1,F4.0,38X)
92200 FORMAT(A5,I1,A14)
C
C Input filespecs
10000 WRITE(*,90000) 'ORAWINI : Interpolate between 2 soundings'
      WRITE(*,90010) '0Input sourcefile 1 (use drivespec) -1/2 '
      READ (*,90020) SRCFL1
      WRITE(*,90010) ' Input sourcefile 2 (use drivespec) -1/2 '
      READ (*,90020) SRCFL2
C
C Read input files
      OPEN(LUFIL,FILE=SRCFL1,STATUS='OLD',FORM='BINARY')
      WRITE(*,90030) SRCFL1,LUFIL,'Opened'
      NSRC1 = 0
10100 NSRC1 = NSRC1 + 1
      READ(LUFIL,END=10200) (DTA(NSRC1,J,1),J=1,7)
      GOTO 10100
C
10200 NSRC1 = NSRC1 - 1
      CLOSE(LUFIL)
      WRITE(*,90030) SRCFL1,LUFIL,'Closed'
C
      OPEN(LUFIL,FILE=SRCFL2,STATUS='OLD',FORM='BINARY')
      WRITE(*,90030) SRCFL2,LUFIL,'Opened'
C
      NSRC2 = 0
10300 NSRC2 = NSRC2 + 1

```

```

      READ(LUFIL,END=10400) (DTA(NSRC2,J,2),J=1,7)
      GOTO 10300

```

C

```

10400 NSRC2 = NSRC2 - 1
      CLOSE(LUFIL)
      WRITE(*,90030) SRCFL2,LUFIL,'Closed'

```

C

C Input interpolation time and destination file

```

11000 WRITE(*,90010) '0Input interpolation YYMMDD -½ '
      READ (*,90040) IYMD0
      WRITE(*,90010) ' Input interpolation HHNN -½ '
      READ (*,90040) IHN0
      WRITE(*,90010) '0Input destination file (use drivespec) -½ '
      READ (*,90020) DSTFIL

```

C

C Read file times

```

      READ(SRCFL1,90050) IYMD1,IHR1,IMIN1
      READ(SRCFL2,90050) IYMD2,IHR2,IMIN2
      IHN1 = IHR1*100 + IMIN1
      IHN2 = IHR2*100 + IMIN2

```

C

C Convert times to minutes after year's start

```

      ITIM1 = MINYR(IYMD1,IHN1)
      ITIM2 = MINYR(IYMD2,IHN2)
      ITIM0 = MINYR(IYMD0,IHN0)
      TIM0 = ITIM0

```

C

C Perform interpolations

```

      NDST = MIN0(NSRC1,NSRC2)
      DO 11500 I = 1, NDST
        TIM1 = ITIM1 + DTA(I,7,1)
        TIM2 = ITIM2 + DTA(I,7,2)
        DTA(I,1,3) = DTA(I,1,1)
        DTA(I,2,3) = XINTRP(TIM1,TIM2,TIM0,DTA(I,2,1),DTA(I,2,2),999.)
        DTA(I,3,3) = XINTRP(TIM1,TIM2,TIM0,DTA(I,3,1),DTA(I,3,2),999.)
        DTA(I,4,3) = XINTRP(TIM1,TIM2,TIM0,DTA(I,4,1),DTA(I,4,2),999.)
        CALL UUVV(DTA(I,5,1),DTA(I,6,1),999.,U1,V1)
        CALL UUVV(DTA(I,5,2),DTA(I,6,2),999.,U2,V2)
        U = XINTRP(TIM1,TIM2,TIM0,U1,U2,999.)
        V = XINTRP(TIM1,TIM2,TIM0,V1,V2,999.)
        CALL WIND(U,V,999.,DTA(I,6,3),DTA(I,5,3))
        DTA(I,7,3) = 0.0

```

```

11500 CONTINUE

```

C

C Output interpolated data

```

      OPEN(LUFIL,FILE=DSTFIL,STATUS='NEW')
      WRITE(*,90030) DSTFIL,LUFIL,'Opened'
      IDATE = IYMD0*100 + IHN0/100
      IMIN = MOD(IHN0,100)
      READ(DSTFIL,90060) CHR

```

```

      LDATE = LOCAL(IDATE,6)
      WRITE(LUFIL,91000) LDATE,IMIN,CHR,NDST,0.,0
      WRITE(LUFIL,91010) IDATE,0.,0.,0.,0.,0.,0.,999.
C
11600 NUMCRD = (NDST - 1)/3 + 1
      LEN      = 0
      CRDNUM   = 0
11700 CRDNUM = CRDNUM + 1
      IF (CRDNUM .LT. NUMCRD) THEN
          DTAFLD = 3
      ELSEIF (CRDNUM .EQ. NUMCRD) THEN
          DTAFLD = MOD(NDST+3,3)
          IF (DTAFLD .EQ. 0) DTAFLD = 3
      ELSEIF (CRDNUM .GT. NUMCRD) THEN
          GOTO 12000
      ENDIF
      LEN = LEN + DTAFLD
      WRITE(DTAfmt,92200) ' (5X,',DTAFLD,'(5I4, I5 )) )'
      WRITE(LUFIL,DTAfmt)
      - ((NINT(DTA(I,J,3)*10.),J=2,7),I=LEN-DTAFLD+1,LEN)
      GOTO 11700
12000 CLOSE(LUFIL)
      WRITE(*,90030) DSTFIL,LUFIL,'Closed'
C
C More ?
      WRITE(*,90010) 'O Do another w/same data (Y/N) ? '
      READ (*,90000) CHR
      IF (CHR .EQ. 'Y') GOTO 11000
      WRITE(*,90010) ' Do another w/different data (Y/N) ? '
      READ (*,90000) CHR
      IF (CHR .EQ. 'Y') GOTO 10000
C
      STOP 'RAWINI -- normal termination'
      END

```

RAWIN Plot module

```

C*****PROGRAM RAWINP
C * * * * *
C 860504 ODGreen RAWIN Plots IBM/PC MS FORTRAN 3.20
C
C      RAWINP processes the RAWINTMP files produced by RAWINJ, and
C schedules the hardcopy output plots.
C
C Subprogram references:
CALLS DFDRV in DFDRV.ASM [odg] Default disk drive
CALLS SEARCH in UTLTYLBR.FOR [odg] Directory search
CALLS DELETE in UTLTYLBR.FOR [odg] File deletion
CALLS GDATE in GDATE.ASM [odg] Get current date

```

```

CALLS XYPLOT in XYPLOT.FOR    [odg]          X,Y|plot generation      *
C                                                                    *
C Notes on satisfying externals:                                     *
C LOOK.ASM required for SEARCH                                       *
C REMOVE.ASM required for DELETE                                     *
C SEARCH.FOR extracted from UTLTYLBR.FOR for separate compilation   *
C DELETE.FOR is contained in &XYPLOT.FOR (odg units for XYPLOT)     *
C $XYPLOT.FOR contains Engel units for plot-10 emulation           *
C                                                                    *
C Link stream:                                                       *
C DFDRV+LOOK+REMOVE+GDATE+RAWINP+SEARCH+                             *
C XYPLOT+&XYPLOT+$XYPLOT,RAWINP.....                               *
C * * * * *                                                           *
PROGRAM RAWINP

INTEGER      TMPDX,TMPNUM,FOUND,TMPLU,PMYSPC,PMXSPC,STNSPC
INTEGER      ZULDAT,LOGDAT,PLTLEN,STATUS,IY,IM,ID,IPP(44)

LOGICAL      BDFLG,ASFLG

REAL         SNDMSG,Y(400),X(400),PMN(30),PMX(30)

CHARACTER    OLDPF*14,NEWPF*14,TMPNAM*12,PMYOPC*2,PMYDES*30
CHARACTER    PMXOPC*2,PMXDES*30,STNDES(12)*1,HDR*45,HD(45)*1
CHARACTER    XLB*30,XL(30)*1,YLB*30,YL(30)*1,TT1*45,TT2*45
CHARACTER    TTL(45,2)*1,TX(195)*1,C(11)*1,MONTH(12)*3

EQUIVALENCE  (TX(1),HD(1)), (TX(46),XL(1)), (TX(76),YL(1)),
-            (TX(106),TTL(1,1))

DATA TMPLU /85/
DATA MONTH /'Jan','Feb','Mar','Apr','May','Jun',
-          'Jul','Aug','Sep','Oct','Nov','Dec'/

DATA IPP /2,0,45,0,0,535,545,0,0,1,0,0,
-         0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
-         0,0,85,520,0,0,115,510,
-         0,0,0,2,0,0,15,75,1,1/

90000 FORMAT('RAWINTMP.',3A1)
90010 FORMAT('ORAWIN_RAWINP :',1X,A12,1X,'Unit:',1X,I4,1X,A)
90020 FORMAT(A2,1X,'vs',1X,A2,3X,I8,1X,I3,'/',I3,1X,'GMT',2X,12A1)
90030 FORMAT('RAWIN(840820)',8X,'Balloon Drift =',1X,A8)
90040 FORMAT('XYPLOT(840616)',11X,'Rundate:',1X,I2,1X,A3,1X,I4)
90045 FORMAT(30A1)
90050 FORMAT(45A1)
90060 FORMAT(1X,4I10)

TMPNUM = 0
CALL DFDRV(TMPDX)

```

```

OLDPF = ' '
NEWPF = 'RAWINPLT.DTA'

```

```

10000 TMPNUM = TMPNUM + 1
      CALL NCOD11(TMPNUM,C)
      WRITE(TMPNAM,90000) (C(J),J=9,11)
      CALL SEARCH(TMPNAM,TMPDX,FOUND)
      IF (FOUND .NE. 0) GOTO 20000

```

```

      OPEN(TMPLU,FILE=TMPNAM,STATUS='OLD',FORM='BINARY')
      WRITE(*,90010) TMPNAM,TMPLU,'Opened'

```

```

      READ(TMPLU) PMYSPC,PMYOPC,PMYDES
      READ(TMPLU) PMXSPC,PMXOPC,PMXDES
      READ(TMPLU) STNSPC,STNDES
      READ(TMPLU) ZULDAT,MINMN,MINMX,BDFLG,ASFLG,SNDMSG
      READ(TMPLU) PMN,PMX

```

```

      PLTLEN = 0

```

```

10100 PLTLEN = PLTLEN + 1
      READ(TMPLU,END=10200) Y(PLTLEN),X(PLTLEN)
      GOTO 10100

```

```

10200 PLTLEN = PLTLEN - 1
      CLOSE(TMPLU)
      WRITE(*,90010) TMPNAM,TMPLU,'Closed'

```

```

      CALL DELETE(TMPNAM,TMPDX,STATUS)
      IF (STATUS .EQ. 0) THEN
        WRITE(*,90010) TMPNAM,TMPLU,'Deleted'
      ELSE
        WRITE(*,90010) TMPNAM,TMPLU,'NOT deleted'
      ENDIF

```

```

      HDR = ' '
      WRITE(HDR,90020) PMXOPC,PMYOPC,ZULDAT,MINMN,MINMX,STNDES
      XLB = PMXDES
      YLB = PMYDES
      READ(HDR,90050) (HD(J),J=1,45)
      READ(XLB,90045) (XL(J),J=1,30)
      READ(YLB,90045) (YL(J),J=1,30)

```

```

      TT1 = ' '
      TT2 = ' '
      IF (BDFLG) THEN
        WRITE(TT1,90030) ' Enabled'
      ELSE
        WRITE(TT1,90030) ' Disabled'
      ENDIF

```

```

CALL GDATE(IY,IM,ID)
WRITE(TT2,90040) ID,MONTH(IM),IY

READ(TT1,90050) (TTL(J,1),J=1,45)
READ(TT2,90050) (TTL(J,2),J=1,45)

IF (ASFLG) THEN
  PMNO = PMX(PMXSPC)
  PMXO = PMN(PMXSPC)
ELSE
  PMNO = PMN(PMXSPC)
  PMXO = PMX(PMXSPC)
ENDIF

CALL XYPLOT(PLTLEN,X,Y,SENDMSG,
-          PMNO,PMXO,PMN(PMYSPC),PMX(PMYSPC),
-          OLDPF,NEWPF,IPP,TX)

GOTO 10000

20000 TMPNAM = 'RAWINJOB.DTA'
OPEN(TMPLU,FILE=TMPNAM,STATUS='OLD')
WRITE(*,90010) TMPNAM,TMPLU,'Opened'

READ(TMPLU,90060) N,I,J,K

IF (I .GT. N) THEN
  CALL DELETE(TMPNAM,TMPDX,STATUS)
  IF (STATUS .EQ. 0) THEN
    WRITE(*,90010) TMPNAM,TMPLU,'Deleted'
  ELSE
    WRITE(*,90010) TMPNAM,TMPLU,'NOT deleted'
  ENDIF
ENDIF

STOP
END

```

RAWIN Plot Module with Clip options

```

C*****PROGRAM RAWINC
C * * * * *
C 860502 ODGreen RAWIN Plots (CLIP) IBM/PC MS FORTRAN 3.20
C
C RAWINC is an enhanced version of RAWINP. It processes the
C RAWINTMP files produced by RAWINJ, and schedules the hardcopy
C output plots.
C The enhancements consist of :

```

```

C      (1) Specification of (up to) 2 plot overlays      *
C      User responds to :                                *
C      NUMBER OF OVERLAYS ? I1                            *
C      MULTIPLE PARAMETER PLOT (Y/N) ? A1                *
C      Count is maintained in IPP(9) [Dash type spec]    *
C      (2) Specification of Clip limits                  *
C      User responds to :                                *
C      ENTER MINIMUM ORDINATE VALUE : E13.2              *
C      ENTER MAXIMUM ORDINATE VALUE : E13.2              *
C      ENTER MINIMUM ABSCISSA VALUE : E13.2              *
C      ENTER MAXIMUM ABSCISSA VALUE : E13.2              *
C      No defaults. Ef.w spec requires decimal point    *
C
C Subprogram references                                  *
CALLS DFDRV   in DFDRV.ASM      [odg]   Default disk drive *
CALLS SEARCH  in UTLTYLBR.FOR   [odg]   Directory search   *
CALLS DELETE  in UTLTYLBR.FOR   [odg]   File deletion      *
CALLS GDATE   in GDATE.ASM      [odg]   Get current date   *
CALLS XYPLOT  in XYPLOT.FOR     [odg]   X,Y plot generation *
C
C Notes on satisfying externals:                        *
C LOOK.ASM required for SEARCH                          *
C REMOVE.ASM required for DELETE                        *
C SEARCH.FOR extracted from UTLTYLBR.FOR for separate compilation *
C DELETE.FOR is contained in &XYPLOT.FOR (odg units for XYPLOT) *
C $XYPLOT.FOR contains Engel units for Plot-10 emulation *
C
C Linkstream:                                           *
C DFDRV+LOOK+REMOVE+GDATE+RAWINC+SEARCH+                *
C XYPLOT+&XYPLOT+$XYPLOT,RAWINC.....                  *
C * * * * *
PROGRAM RAWINC

INTEGER      TMPDX,TMPNUM,FOUND,TMPLU,PMYSPC,PMXSPC,STNSPC
INTEGER      ZULDAT,LOCDAT,PLTLEN,STATUS,IY,IM,ID,IPP(44)

LOGICAL      BDFLG,ASFLG,MPPLOT

REAL         SNDMSG,Y(400),X(400),PMN(30),PMX(30)

CHARACTER    OLDPF*14,NEWPF*14,TMPNAM*12,PMYOPC*2,PMYDES*30
CHARACTER    PMXOPC*2,PMXDES*30,STNDES(12)*1,HDR*45,HD(45)*1
CHARACTER    XLB*30,XL(30)*1,YLB*30,YL(30)*1,TT1*45,TT2*45
CHARACTER    TTL(45,2)*1,TX(195)*1,C(11)*1,MONTH(12)*3,ANS*1

EQUIVALENCE  (TX(1),HD(1)), (TX(46),XL(1)), (TX(76),YL(1)),
              (TX(106),TTL(1,1))

DATA TMPLU /85/
DATA MONTH /'Jan','Feb','Mar','Apr','May','Jun',

```

- 'Jul','Aug','Sep','Oct','Nov','Dec'/'

DATA IPP /2,0,45,0,0,535,545,0,0,1,0,0,
 - 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 - 0,0,85,520,0,0,115,510,
 - 0,0,0,2,0,0,15,75,0,1/

90000 FORMAT('RAWINTMP.',3A1)
 90010 FORMAT('ORAWIN_RAWINP :',1X,A12,1X,'Unit:',1X,I4,1X,A)
 90020 FORMAT(A2,1X,'vs',1X,A2,1X,I1,1X,I8,1X,I3,'/',I3,1X,'GMT',2X,12A1)
 90030 FORMAT('RAWIN(840820)',8X,'Balloon Drift =',1X,A8)
 90040 FORMAT('XYPLOT(840616)',11X,'Rundate:',1X,I2,1X,A3,1X,I4)
 90045 FORMAT(30A1)
 90050 FORMAT(45A1)
 90060 FORMAT(1X,4I10)

TMPNUM = 0
 CALL DFDRV(TMPDX)
 OLDPF = ' '
 NEWPF = 'RAWINPLT.DTA'

WRITE(*,'(A,*)') 'ONUMBER OF OVERLAYS ?'
 READ (*,'(I1)') NOVLAY
 WRITE(*,'(A,*)') ' MULTIPLE PARAMETER PLOT (Y/N) ? '
 READ (*,'(A)') ANS
 MPLOT = ANS .EQ. 'Y'

10000 TMPNUM = TMPNUM + 1
 CALL NCODI1(TMPNUM,C)
 WRITE(TMPNAM,90000) (C(J),J=9,11)
 CALL SEARCH(TMPNAM,TMPDX,FOUND)
 IF (FOUND .NE. 0) GOTO 20000

OPEN(TMPLU,FILE=TMPNAM,STATUS='OLD',FORM='BINARY')
 WRITE(*,90010) TMPNAM,TMPLU,'Opened'

READ(TMPLU) PMYSPC,PMYOPC,PMYDES
 READ(TMPLU) PMXSPC,PMXOPC,PMXDES
 READ(TMPLU) STNSPC,STNDES
 READ(TMPLU) ZULDAT,MINMN,MINMX,BDFLG,ASFLG,SNMSG
 READ(TMPLU) PMN,PMX

WRITE(*,'(A,*)') 'OENTER MINIMUM ORDINATE VALUE : '
 READ (*,'(E13.2)') PMN(PMYSPC)
 WRITE(*,'(A,*)') ' ENTER MAXIMUM ORDINATE VALUE : '
 READ (*,'(E13.2)') PMX(PMYSPC)

WRITE(*,'(A,*)') 'OENTER MINIMUM ABSCISSA VALUE : '
 READ (*,'(E13.2)') PMN(PMXSPC)
 WRITE(*,'(A,*)') ' ENTER MAXIMUM ABSCISSA VALUE : '

```
READ (*,'(E13.2)') PMX(PMXSPC)
```

```
PLTLEN = 0
```

```
10100 PLTLEN = PLTLEN + 1
```

```
READ(TMPLU,END=10200) Y(PLTLEN),X(PLTLEN)
```

```
IF (X(PLTLEN) .LT. PMN(PMXSPC)) X(PLTLEN) = PMN(PMXSPC)
```

```
IF (X(PLTLEN) .GT. PMX(PMXSPC)) X(PLTLEN) = PMX(PMXSPC)
```

```
IF (Y(PLTLEN) .LT. PMN(PMYSPC)) THEN
```

```
    PLTLEN = PLTLEN - 1
```

```
    GOTO 10100
```

```
ELSEIF (Y(PLTLEN) .GT. PMX(PMYSPC)) THEN
```

```
    GOTO 10200
```

```
ENDIF
```

```
GOTO 10100
```

```
10200 PLTLEN = PLTLEN - 1
```

```
CLOSE(TMPLU)
```

```
WRITE(*,90010) TMPNAM,TMPLU,'Closed'
```

```
CALL DELETE(TMPNAM,TMPDX,STATUS)
```

```
IF (STATUS .EQ. 0) THEN
```

```
    WRITE(*,90010) TMPNAM,TMPLU,'Deleted'
```

```
ELSE
```

```
    WRITE(*,90010) TMPNAM,TMPLU,'NOT deleted'
```

```
ENDIF
```

```
HDR = ' '
```

```
IF (IPP(9) .EQ. 0) THEN
```

```
    WRITE(HDR,90020) PMXOPC,PMYOPC,IPP(9),ZULDAT,MINMN,MINMX,STNDES
```

```
    XLB = PMXDES
```

```
    IF (MPPLOT) XLB = ' '
```

```
    YLB = PMYDES
```

```
    READ(HDR,90050) (HD(J),J=1,45)
```

```
    READ(XLB,90045) (XL(J),J=1,30)
```

```
    READ(YLB,90045) (YL(J),J=1,30)
```

```
ENDIF
```

```
TT1 = ' '
```

```
TT2 = ' '
```

```
IF (NOVLAY .EQ. 0) THEN
```

```
    IF (BDFLG) THEN
```

```
        WRITE(TT1,90030) ' Enabled'
```

```
    ELSE
```

```
        WRITE(TT1,90030) ' Disabled'
```

```
    ENDIF
```

```
    CALL GDATE(IY,IM,ID)
```

```
    WRITE(TT2,90040) ID,MONTH(IM),IY
```

```
ELSE
```

```
    IF (IPP(9) .EQ. 1)
```

```

-      WRITE(TT1,90020) PMXOPC,PMYOPC,IPP(9),ZULDAT,
-      MINMN,MINMX,STNDES
-      IF (IPP(9) .EQ. 2)
-      WRITE(TT2,90020) PMXOPC,PMYOPC,IPP(9),ZULDAT,
-      MINMN,MINMX,STNDES
ENDIF
READ(TT1,90050) (TTL(J,1),J=1,45)
READ(TT2,90050) (TTL(J,2),J=1,45)

```

```

IF (ASFLG) THEN
  PMNO = PMX(PMXSPC)
  PMXO = PMN(PMXSPC)
ELSE
  PMNO = PMN(PMXSPC)
  PMXO = PMX(PMXSPC)
ENDIF

```

```

IF (IPP(9) .EQ. NOVLAY) IPP(44) = 1
IF (IPP(9) .NE. NOVLAY) IPP(44) = 0
CALL XYPLOT(PLTLEN,X,Y,SNDMSG,
-      PMNO,PMXO,PMN(PMYSPC),PMX(PMYSPC),
-      OLDPF,NEWPF,IPP,TX)
IF (IPP(44) .EQ. 1) THEN
  IPP(9) = 0
  OLDPF = ' '
ELSE
  IPP(9) = IPP(9) + 1
  OLDPF = NEWPF
ENDIF
GOTO 10000

```

```

20000 TMPNAM = 'RAWINJOB.DTA'
OPEN(TMPLU,FILE=TMPNAM,STATUS='OLD')
WRITE(*,90010) TMPNAM,TMPLU,'Opened'

```

```

READ(TMPLU,90060) N,I,J,K

```

```

IF (I .GT. N) THEN
  CALL DELETE(TMPNAM,TMPDX,STATUS)
  IF (STATUS .EQ. 0) THEN
    WRITE(*,90010) TMPNAM,TMPLU,'Deleted'
  ELSE
    WRITE(*,90010) TMPNAM,TMPLU,'NOT deleted'
  ENDIF
ENDIF

```

```

STOP
END

```

Rawinsonde Subprogram UnitsBubble sort

C*****SUBROUTINE BSORT(A,N,S)

* * * * *

* 15 July 1984 Green

* * * * *

* Perform bubble sort on real array

* * * * *

* Entry:

* A [R] array of values to be sorted

* N [I] number of elements in A

* S [I] sort switch :

* = 0 ascending order

* $\frac{1}{2}$ 0 descending order

* * * * *

* Exit:

* A sorted according to S

* * * * *

SUBROUTINE BSORT(A,N,S)

INTEGER N,S,M

REAL A(N),HOLD

LOGICAL SWITCH,SWAP

*Begin BSORT

M = N - 1

10 IF (M .LT. 1) RETURN

SWITCH = .FALSE.

DO 20 I = 1, M

IF (S .EQ. 0) THEN

SWAP = A(I) .GT. A(I+1)

ELSE

SWAP = A(I) .LT. A(I+1)

ENDIF

IF (SWAP) THEN

SWITCH = .TRUE.

HOLD = A(I)

A(I) = A(I+1)

A(I+1) = HOLD

ENDIF

20 CONTINUE

M = M - 1

IF (SWITCH) GOTO 10

*Terminate BSORT

RETURN
END

File deletion

```

C-----
C*****SUBROUTINE DELETE (FILNAM,DRIVE,STATUS)
C-----
C 15 MAY 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
C      DELETE FILE ON SPECIFIED DRIVE. FILE MUST BE UNOPENED.
C ENTRY
C      FILNAM [C*12]  FILE NAME (NO DRIVE SPEC)  EG: FILENAME.EXT
C                      (LEFT-JUSTIFIED WITH TRAILING BLANKS)
C      DRIVE  [I]    DISK DRIVE NUMBER (1=A,2=B,ETC. 0=DEFAULT)
C                      ***IT IS THE USER'S RESPONSIBILITY TO ENSURE THAT
C                      THE DISK DRIVE NUMBER IS VALID.
C EXIT
C      STATUS [I]    = 0    FILE DELETED
C                      = 255 FILE NOT DELETED
C SUBPROGRAM REFERENCES
CALLS REMOVE
C-----
      SUBROUTINE DELETE (FILNAM,DRIVE,STATUS)
$DEBUG
      CHARACTER      FILNAM*12,FN(8)*1,FX(3)*1,F(12)*1,FCB*37
      INTEGER        DRIVE,STATUS
C
91000 FORMAT(12A1)
91040 FORMAT(' ',8A1,3A1,25(' '))
C
C      DECODE FILE NAME. PAD FIELDS W/BLANKS WHERE NECESSARY
1000 READ(FILNAM,91000) F
      DO 1010 I = 1, 8
          IF (F(I) .EQ. '.') THEN
              GOTO 1020
          ELSE
              FN(I) = F(I)
          ENDIF
1010 CONTINUE
1020 DO 1030 J = 1, 8
          FN(J) = ' '
1030 CONTINUE
C      DECODE FILE EXTENT
      I = I + 1
      DO 1040 J = I, I + 2
          FX(J-I+1) = F(J)
1040 CONTINUE
C      BUILD FCB, DO DELETE, THEN QUIT
      FCB = ' '

```

```

WRITE(FCB,91040) FN,FX
CALL REMOVE(FCB,DRIVE,STATUS)
RETURN
END

```

Drive number

```

C*****INTEGER FUNCTION DXNUM (CHR)
C-----
C 24 APRIL 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
C      DXNUM CONVERTS THE DRIVE LETTER TO A NUMBER.
C ENTRY
C      CHR  [C*1]    DRIVE LETTER
C EXIT
C      DXNUM [I]     DRIVE NUMBER
C                      IF DRIVE LETTER NOT A-Z (UPPER OR LOWER CASE),
C                      DXNUM RETURNED AS 0
C-----
C      INTEGER FUNCTION DXNUM (CHR)
C      CHARACTER          CHR*1
C
C      DXNUM = 0
C      IF ( (ICHAR(CHR) .GE. 65) .AND. (ICHAR(CHR) .LE. 90))
C      -   DXNUM = ICHAR(CHR) - 64
C      IF ( (ICHAR(CHR) .GE. 97) .AND. (ICHAR(CHR) .LE. 122))
C      -   DXNUM = ICHAR(CHR) - 96
C      RETURN
C      ,END

```

U,V wind components

```

C*****SUBROUTINE UUVV(W,D,M,U,V)
C*****
C U,V wind components 840715 Green
C      W [R]  wind speed [m/s]
C      D [R]  wind direction [deg]
C      M [R]  value assigned to a missing datum
C      U [R]  eastward wind component [m/s] or M if W or D = M
C      V [R]  northward wind component [m/s] or M if W or D = M
C*****
C      SUBROUTINE UUVV(W,D,M,U,V)
C      REAL W,D,M,U,V,A,RAD
C      DATA RAD/0.01745/
C
C      IF ( (W .EQ. M) .OR. (D .EQ. M) ) THEN
C          U = M
C          V = M
C      ELSE

```

```

      A = (270. - D) * RAD
      U = W * COS(A)
      V = W * SIN(A)
ENDIF

```

```

RETURN
END

```

Hypsometric evaluation

```

C*****REAL FUNCTION DZGPM(T1,D1,P1,T2,D2,P2,M)
* * * * *
* Height (gpm) between 2 levels (Saucier - 41) 840715 Green
CALLS TVIRT to calculate the virtual temperature
*
* Mean virtual temperature taken to be simple arithmetic average
* If dewpoints missing, virtual temp taken as ambient temp
* * * * *
      REAL FUNCTION DZGPM(T1,D1,P1,T2,D2,P2,M)
      REAL T1,D1,P1,T2,D2,P2,M,TV1,TV2,TVIRT,TV,K
      DATA K /29.28980/

      IF ( (T1.EQ.M) .OR. (P1.EQ.M) .OR. (T2.EQ.M) .OR. (P2.EQ.M)) THEN
        DZGPM = M
      ELSE
        IF (D1 .NE. M) THEN
          TV1 = TVIRT(T1,D1,P1,M)
        ELSE
          TV1 = T1
        ENDIF
        IF (D2 .NE. M) THEN
          TV2 = TVIRT(T2,D2,P2,M)
        ELSE
          TV2 = T2
        ENDIF
        TV = (TV1 + TV2) / 2
        DZGPM = (K * TV) * ALOG(P1/P2)
      ENDIF

      RETURN
      END

```

Equivalent potential temperature

```

C*****REAL FUNCTION TEQPOT(T,D,P,M)
* * * * *
* Equivalent pot temp - Rossby def (Saucier - 14) 840715 Green
CALLS TPOT

```

CALLS MIXRAT

```

REAL FUNCTION TEQPOT(T,D,P,M)
REAL T,D,P,M,TP,TPOT,R,MIXRAT,L,CP,TA
DATA L,CP/540.,0.240/

IF ( (T.EQ.M) .OR. (D.EQ.M) .OR. (P.EQ.M) ) THEN
  TEQPOT = M
ELSE
  TP = TPOT(T,P,M)
  TA = T + 273.16
  R = MIXRAT(D,P,M)
  TEQPOT = TP * EXP( (L*R) / (CP*TA) )
ENDIF

RETURN
END

```

Equivalent temperature

```

C*****REAL FUNCTION TEQ(T,D,P,M)
*****
* Equivalent temp - Rossby definition (Saucier - 14) 840715 Green *
CALLS MIXRAT
*****

```

```

REAL FUNCTION TEQ(T,D,P,M)
REAL T,D,P,M,R,MIXRAT,L,CP,TA
DATA L,CP /540.0,0.240/

IF ( (T.EQ.M) .OR. (D.EQ.M) .OR. (P.EQ.M) ) THEN
  TEQ = M
ELSE
  R = MIXRAT(D,P,M)
  TA= T + 273.16
  TEQ = TA * EXP( (L*R) / (CP*TA))
ENDIF

RETURN
END

```

Potential Temperature

```

C*****REAL FUNCTION TPOT(T,P,M)
*****
* Potential temperature (Saucier - 12) 840715 Green *
* Potential temperature (Saucier - 12) *

```

```

REAL FUNCTION TPOT(T,P,M)
REAL T,P,M,K

IF ( (T.EQ.M) .OR. (P.EQ.M) ) THEN
  TPOT = M
ELSE
  K = 2./7.
  TPOT = (T + 273.16) * (1000./P) ** K
ENDIF

RETURN
END

```

Virtual temperature

```

C*****REAL FUNCTION TVIRT(T,D,P,M)
*****
* Virtual temperature (Saucier - 12) 840715 Green
* T [R] temperature [C]
* D [R] dewpoint [C]
* P [R] pressure [mb]
* M [R] value assigned to a missing datum
CALLS SPFHUM
*****

```

```

REAL FUNCTION TVIRT(T,D,P,M)
REAL T,D,P,M,Q,SPFHUM

```

*Begin TVIRT

```

IF ( (T.EQ.M) .OR. (D.EQ.M) .OR. (P.EQ.M) ) THEN
  TVIRT = M
ELSE
  Q = SPFHUM(D,P,M)
  TVIRT = (T+273.16) * (1 + 0.61*Q)
ENDIF

```

*Terminate TVIRT

```

RETURN
END

```

Relative Humidity

```

C*****REAL FUNCTION RELHUM(T,D,P,M)
*****
* Relative humidity (Saucier 11) 840715 Green

```

```

*      T [R]  temperature [C]
*      D [R]  dewpoint [C]
*      P [R]  pressure [mb]
*      M [R]  value assigned to a missing datum
*      RELHUM [R] relative humidity [%] or M if T,D,or P = M
CALLS MIXRAT
* * * * *
REAL FUNCTION RELHUM(T,D,P,M)
REAL T,D,P,M,R,RS,MIXRAT
IF ( (T.EQ.M) .OR. (D.EQ.M) .OR. (P.EQ.M) ) THEN
  RELHUM = M
ELSE
  R = MIXRAT(D,P,M)
  RS = MIXRAT(T,P,M)
  RELHUM = (R/RS) * 100.0
ENDIF
RETURN
END

```

Mixing Ratio

```

C*****REAL FUNCTION MIXRAT(D,P,M)
* * * * *
* Mixing ratio (Saucier - 10) 840715 Green
*      D [R]  dewpoint [C] (or T temp [C] for saturation value)
*      P [R]  pressure [mb]
*      M [R]  value assigned to a missing datum
*      MIXRAT [R] mixing ratio [g/g] or M if D or P = M
CALLS VAPOR
* * * * *
REAL FUNCTION MIXRAT(D,P,M)
REAL D,P,M,FW,E,VAPOR
DATA FW/1.0/

IF ( (D.EQ.M) .OR. (P.EQ.M) ) THEN
  MIXRAT = M
ELSE
  E = VAPOR(D,M)
  MIXRAT = (0.62197 * FW * E) / (P - FW*E)
ENDIF

RETURN
END

```

Specific humidity

```

C*****REAL FUNCTION SPFHUM(D,P,M)
* * * * *

```

```

* Specific humidity (Saucier - 10) 840715 Green
*   D [R] dewpoint [C]
*   P [R] pressure [mb]
*   M [R] value assigned to a missing datum
*   SPFHUM [R] specific humidity [g/g] or M if D or P = M
CALLS VAPOR
* * * * *
REAL FUNCTION SPFHUM(D,P,M)
REAL D,P,M,FW,E,VAPOR
DATA FW/1.0/

IF ( (D.EQ.M) .OR. (P.EQ.M) ) THEN
  SPFHUM = M
ELSE
  E = VAPOR(D,M)
  SPFHUM = (0.62197*FW*E) / (P - (0.37803*FW*E))
ENDIF

RETURN
END

```

Vapor pressure

```

C*****REAL FUNCTION VAPOR(T,M)
* * * * *
* Vapor pressure over water (Tabata, 1973) 840715 Green
*   T [R] temperature [C] (or D dewpoint for actual vapor pres)
*   M [R] value assigned to a missing datum
*   VAPOR [R] sat. vapor pressure [mb] or M if T=M
* * * * *
REAL FUNCTION VAPOR(T,M)
REAL T,M
DOUBLE PRECISION TT,VP,A1,A2,A3
DATA A1,A2,A3/8.42926609,-1.82717843,-0.071208271/

IF (T .EQ. M) THEN
  VAPOR = M
ELSE
  TT = DBLE((10.**3)/(T+273.16))
  VP = 10. ** (A1 + A2*TT + A3 * (TT**2))
  VAPOR = SNGL(VP)
ENDIF

RETURN
END

```

Encode an integer *1 array

C*****SUBROUTINE NCODI1(I,C)

C Version 1.00 (10 October 1983)

by Orville D. Green

C Update

C Description:

C NCODI1 encodes an integer variable into a character array
 C 11 characters in length. The leftmost element contains
 C a minus sign for negative numbers, or a blank for positive
 C numbers. Thereafter, the array is filled with leading zeros,
 C followed by the encoded integer digits.

C Arguments:

C I [integer] : integer to be encoded

C C [character *1] : (returned) character *1 encoding [DIM(11)]

SUBROUTINE NCODI1 (I,C)

\$DEBUG

C argument declarations
 C INTEGER I
 C CHARACTER *1 C(11)

C local declarations
 C CHARACTER *11 K

*Begin NCODI1

```

      K = ' '
      WRITE(K,'(I11)') I
      READ(K,'(11A1)') C
      DO 10 J = 2, 11
        IF(C(J) .EQ. ' ') C(J) = '0'
        IF(C(J) .EQ. '-') THEN
          C(1) = '-'
          C(J) = '0'
        ENDIF
      10 CONTINUE

```

*Terminate NCODI1

99999 RETURN

END

Average RAWIN station values

```

C*****REAL FUNCTION RWAVG(SCC,VAL,VALLN,VALMSG)
* * * * *
* Average RAWIN station values                                840820 Green *
CALLS BITVAL
* * * * *
      REAL FUNCTION RWAVG(SCC,VAL,VALLN,VALMSG)

```

```

$INCLUDE: 'RAWINJOB.DCL'

```

```

      INTEGER SCC,VALLN,NSCC,BITVAL
      REAL VAL(VALLN),VALMSG,VBAR

```

```

*Begin RWAVG

```

```

      VBAR = 0.0
      NSCC = 0
      DO 10000 I = 1, VALLN
        IF (BITVAL(SCC,I-1) .EQ. SET) THEN
          NSCC = NSCC + 1
          IF(VAL(I) .NE. VALMSG) THEN
            VBAR = VBAR + VAL(I)
          ELSE
            RWAVG = VALMSG
            GOTO 89000
          ENDIF
        ENDIF
      ENDIF
      10000 CONTINUE
      RWAVG = VBAR / FLOAT(NSCC)

```

```

*Terminate RWAVG
      89000 RETURN
      END

```

RAWIN sounding file dump

```

C*****SUBROUTINE RWDMP
* * * * *
* RAWIN sounding file dump                                840820 Green *
CALLS BITVAL
CALLS SEARCH
* * * * *

```

```

      SUBROUTINE RWDMP

```

```

$INCLUDE: 'RAWINJOB.DCL'

```

```

INTEGER    LP,BITVAL
LOGICAL    REQ,AVL
CHARACTER  CARD*80

```

```
DATA LP /6/
```

```

90000 FORMAT('RW',I6,'.',I2,A1)
90010 FORMAT('RW',I6,'.0',I1,A1)
90020 FORMAT(A2,A12)
90030 FORMAT(/,1X,'RAWIN_RWDMPs:',1X,A14,1X,'Unit:',I4,1X,A)
90040 FORMAT(/,1X,'RAWIN_RWDMPs:',1X,A14,1X,'NOT found')
90050 FORMAT('1',26X,'RAWIN dump of sounding file ',A12,1X,':')
90060 FORMAT(26X,A80)
90070 FORMAT(/,1X,'RAWIN_RWDMPs file I/O error : ',I8)

```

```
*Begin RWDMPs
```

```

OPEN(LP,FILE='LPT1:')
WRITE(*,90030) 'LPT1:           ',LP,'Opened'
SNDGMT = GETDAT(GETNUM)

```

```
C Dump only if required and available
```

```

10000 DO 19000 STNNUM = 1, STNMAX
      REQ = BITVAL(STNREQ,STNNUM-1) .EQ. SET
      AVL = BITVAL(GETSTN(GETNUM),STNNUM-1) .EQ. SET
      IF ( (.NOT. REQ) .OR. (.NOT. AVL) ) GOTO 19000

```

```
C Build sounding filespec
```

```

      IHR = MOD(SNDGMT,100)
      IF (IHR .GE. 10) THEN
        WRITE(SNDNAM,90000) SNDGMT/100,IHR,STID01(STNNUM)
      ELSE
        WRITE(SNDNAM,90010) SNDGMT/100,IHR,STID01(STNNUM)
      ENDIF
      WRITE(SNDFIL,90020) DRIVE(SNDDX),SNDNAM

```

```
C Open sounding file. Error stop if not disk-resident and not pause flag
```

```

11000 CALL SEARCH(SNDNAM,SNDDX,FOUND)
      IF (FOUND .EQ. 0) THEN
        OPEN(SNDLU,FILE=SNDFIL,STATUS='OLD')
        WRITE(*,90030) SNDFIL,SNDLU,'Opened'
      ELSE
        WRITE(*,90040) SNDFIL
        IF (PSFLG) THEN
          PAUSE 'Exchange diskettes'
          GOTO 11000
        ELSE
          GOTO 88000
        ENDIF

```

ENDIF

C Dump the file

WRITE(LP,90050) SNDNAM

```
DO 17000 I = 1, 200
```

CARD = ' '

```
READ(SNDLU,90060,END=18000,ERR=15000) CARD
```

GOTO 16000

C Clear -1252 error code (variable length records). Else error stop.

```
15000      IERR = IGETER(SNDLU)
```

```
IF (IERR .EQ. 1252) THEN
```

CALL ICLRER(SNDLU)

GOTO 16000

ELSE

```
WRITE(*,90070) IERR
```

GOTO 88000

ENDIF

C Write out the record

```
16000      WRITE(LP,90060) CARD
```

17000 CONTINUE

C Close the sounding file

18000 CLOSE(SNDLU)

```
WRITE(*,90030) SNOFIL,SNDLU,'Closed'
```

19000 CONTINUE

C Close line printer

CLOSE(LP)

```
WRITE(*,90030) 'LPT1:      ',LP,'Closed'
```

GOTO 89000

ABEND

```
88000 STOP 'RAWIN_RWDMPS abnormal ending'
```

*Terminate RWDMPs

89000 RETURN

END

Set RAWIN processing flags

C*****SUBROUTINE RWFLGI

* * * * *

```
* Set RAWIN processing flags                                     840820 Green *
```

CALLS BITVAL *

CALLS RWOPCI *

* * * * *

SUBROUTINE RWFLGI

```
$INCLUDE: 'RAWINTSK.DCL'
```

```
INTEGER    TSTBIT,BITVAL,SFNUM
CHARACTER  STATUS*2,BELL*1
```

```
90000 FORMAT(A)
90010 FORMAT(5X,A2,1X,A30,1X,'Status = ',A2)
90020 FORMAT(A,A,*)
```

```
*Begin RWFLGI
```

```
BELL  = CHAR(7)
TSTBIT = BIT0
```

```
C Output current flag status
```

```
10000 WRITE(*,90000) '0Current Processing Flag Status : '
      WRITE(*,90000) ' '
      DO 10010 I = 1, SFLEN
        IF (BITVAL(SFBIT(I),TSTBIT) .EQ. CLR) THEN
          IF (RWFLGX(I)) THEN
            STATUS = 'ON'
          ELSE
            STATUS = 'OF'
          ENDIF
          WRITE(*,90010) SFOPC(I),SFDES(I),STATUS
        ENDIF
      10010 CONTINUE
```

```
C Input flag to enable/disable
```

```
CALL RWOPCI(SFLEN,SFOPC,SFDES,SFBIT,SFTTL,TSTBIT,SFNUM)
```

```
C Enable or disable the flag and redisplay status, unless exit indicated
```

```
      IF (SFNUM .NE. 0) THEN
10200  IF (RWFLGX(SFNUM)) THEN
          STATUS = 'ON'
        ELSE
          STATUS = 'OF'
        ENDIF
        WRITE(*,90010) SFOPC(SFNUM),SFDES(SFNUM),STATUS
        WRITE(*,90020) ' Enable (ON) or Disable (OF) ? ',BELL
        READ (*,90000) STATUS
        IF ( (STATUS .NE. 'ON') .AND. (STATUS .NE. 'OF') ) GOTO 10200
        RWFLGX(SFNUM) = STATUS .EQ. 'ON'
        GOTO 10000
      ENDIF
```

```
*Terminate RWFLGI
```

```
89000 RETURN
```

END

Fetch sounding data for all required stations

```

C*****SUBROUTINE RWGETA(STNREQ,PSEFLG)
* * * * *
* 15 July 1984  Green
*
*      Fetch sounding data for all required stations
* * * * *

```

SUBROUTINE RWGETA(STNREQ,PSEFLG)

\$INCLUDE:'RAWINGET.DCL'

\$INCLUDE:'RWDTA.DCL'

```

      INTEGER    STNREQ, BITVAL
      LOGICAL    PSEFLG

```

*Begin RWGETA

C Assume all data missing

```

10000 DO 10010 I = 1, SNDMAX*SNDPRM*STNMAX
      PDX(I) = SNDMSG

```

10010 CONTINUE

C Fetch all stations

20000 DO 20020 STNNUM = 1, STNMAX

20005 CALL RWGETS

IF (SNDLEN .EQ. -1) THEN

IF (PSEFLG) THEN

PAUSE 'Exchange sounding file diskettes'

GOTO 20005

ELSE

GOTO 88000

ENDIF

ELSEIF (SNDLEN .EQ. -2) THEN

GOTO 88000

ENDIF

NX(STNNUM) = SNDLEN

DO 20010 I = 1, NX(STNNUM)

PRX(I,STNNUM) = PRC(I)

TEX(I,STNNUM) = TEC(I)

TDX(I,STNNUM) = TDC(I)

WSX(I,STNNUM) = WSC(I)

WDX(I,STNNUM) = WDC(I)

TMX(I,STNNUM) = TMC(I)

20010 CONTINUE

```
20020 CONTINUE
      GOTO 89000
```

```
***ABEND***
```

```
88000 STOP 'RAWIN_RWGETA abnormal ending'
```

```
*Terminate RWGETA
89000 RETURN
      END
```

Get fetch list

```
C*****SUBROUTINE RWGETL(PSEFLG,TTINT)
* * * * *
* RAWIN - Get fetch list for user-specified period      840810 Green *
* * * * *

      SUBROUTINE RWGETL(PSEFLG,TTINT)

$INCLUDE:'RWMSC.DCL'
$INCLUDE:'RWIDX.DCL'
$INCLUDE:'RWSND.DCL'
$INCLUDE:'RWSTN.DCL'

      LOGICAL      PSEFLG,FIRST,SWITCH,SPANHR
      INTEGER      TTINT,DXNUM,ZULU,BITVAL,IVSCC,ISCC,MINYR
      INTEGER      YMD,HNS,HN,TI
      INTEGER      SRTLST(150),SRTSTN(150),SRTMIN(150),SRTLLEN,SRTMAX
      INTEGER      BTMPTR,TOPPTR
      CHARACTER     BELL *1, ANS *1, SCCTXT(12) *1

      DATA  SRTMAX /150/

90000 FORMAT(A,A,*)
90010 FORMAT(A)
90015 FORMAT(A,1X,I2,1X,A)
90020 FORMAT(I8,1X,I8,1X,A)
90030 FORMAT(/,1X,I4,A)
90040 FORMAT(1X,I2,'-',I8,1X,I3,'/',I3,2X,I2A1,2X,*)
90045 FORMAT(1X,I2,'-',I8,*)
90046 FORMAT(1X,A1,1X,I3,1X,*)

*Begin RWGETL

      BELL = CHAR(7)

C Input index drive
10000 WRITE(*,90000) '0Input INDEX file drive letter or ¼CR½:',BELL
```

```

READ (*,90010)  ANS
IDXDX = DXNUM(ANS)
IF (IDXDX .EQ. 0) THEN
  CALL DFDRV(I)
  IDXDX = I
ENDIF

```

C Input the period to process

```

10500 WRITE(*,90010) '0Enter BEGINNING and ENDING dates, TIMEZONE'
WRITE(*,90000) ' e.g., YYMMDDHH,YYMMDDHH,L (L if MDT) : ',BELL
READ (*,90020)  BEGDAT,ENDDAT,ANS

```

C Convert times to zulu

```

IF (ANS .EQ. 'L') THEN
  BEGDAT = ZULU(BEGDAT,MDTGMT)
  ENDDAT = ZULU(ENDDAT,MDTGMT)
ENDIF

```

C Access the index file to get soundings available

```

11000 CALL RWGETX

```

C Process RWGETX returned errors

```

IF (IDXLEN .EQ. 0) THEN
  WRITE(*,90010) '0No soundings available for requested period'
  WRITE(*,90000) ' Re-specify period (Y/N) ? ',BELL
  READ (*,90010)  ANS
  IF (ANS .EQ. 'Y') THEN
    GOTO 10500
  ELSE
    GOTO 88000
  ENDIF
ELSEIF (IDXLEN .EQ. -1) THEN
  IF (PSEFLG) THEN
    GOTO 10000
  ELSE
    GOTO 88000
  ENDIF
ELSEIF (IDXLEN .EQ. -2) THEN
  GOTO 88000
ENDIF

```

C No RWGETX errors. Build fetch lists. If the caller isn't concerned
C with the time period between soundings, simply transfer the index
C lists to the fetch lists.

```

20000 IF (TTINT .LT. 0) THEN
  DO 21000 IDXNUM = 1, IDXLEN
    GETDAT(IDXNUM) = IDXDAT(IDXNUM)
    GETSTN(IDXNUM) = CLR
  DO 20500 STNNUM = 1, STNMAX
    GETMIN(IDXNUM,STNNUM) = IDXMIN(IDXNUM,STNNUM)

```

```

        IF (IDXMIN(IDXNUM,STNNUM) .NE. MSGSND) THEN
            CALL SETBIT(GETSTN(IDXNUM),STNNUM-1)
        ENDIF
20500     CONTINUE
21000     CONTINUE
        GETLEN = IDXLEN

C Otherwise, calculate the number of minutes after the start of the
C year to the release time for each sounding in the index lists, and
C put on the sort lists
        ELSE
25000     DO 25010 I = 1, GETMAX
            DO 25010 J = 1, STNMAX
                GETMIN(I,J) = MSGSND
25010     CONTINUE

        SRTLEN = 0
        DO 25100 IDXNUM = 1, IDXLEN
            DO 25100 STNNUM = 1, STNMAX
                IF (IDXMIN(IDXNUM,STNNUM) .NE. MSGSND) THEN
                    SRTLEN = SRTLEN + 1
                    IF (SRTLEN .GT. SRTMAX) THEN
                        SRTLEN = SRTMAX
                        WRITE(*,90010) 'ORAWIN_RWGETL sort lists full'
                        GOTO 25150
                    ENDIF
                    SRTSTN(SRTLEN) = STNNUM
                    SRTMIN(SRTLEN) = IDXMIN(IDXNUM,STNNUM)
                    YMD = IDXDAT(IDXNUM) / 100
                    TI = 0
                    IF (SRTMIN(SRTLEN) .LT. 0) THEN
                        SRTMIN(SRTLEN) = SRTMIN(SRTLEN) + 60
                        TI = -1
                    ENDIF
                    HNS = MOD(IDXDAT(IDXNUM),100)*10000 + SRTMIN(SRTLEN)*100
                    CALL TIMINC(YMD,HNS,TI,3)
                    HN = HNS / 100
                    SRTLST(SRTLEN) = MINYR(YMD,HN)
                ENDIF
25100     CONTINUE

C Sort the list (ascending order) -- keep track of stations, minutes
25150     M = SRTLEN
25200     M = M - 1
        SWITCH = .FALSE.
        DO 25300 I = 1, M
            IF (SRTLST(I) .GT. SRTLST(I+1)) THEN
                IHOLDL = SRTLST(I)
                IHOLDS = SRTSTN(I)
                IHOLDM = SRTMIN(I)

```

```

        SRTLST(I) = SRTLST(I+1)
        SRTSTN(I) = SRTSTN(I+1)
        SRTMIN(I) = SRTMIN(I+1)
        SRTLST(I+1) = IHOLDL
        SRTSTN(I+1) = IHOLDS
        SRTMIN(I+1) = IHOLDM
        SWITCH = .TRUE.
    ENDIF
25300    CONTINUE
        IF (SWITCH) GOTO 25200

C Build the fetch list according to the time period between soundings
    GETLEN = 0
    BTMPTR = 1
    TOPPTR = 1

C Add date at bottom pointer to fetch list
25400    GETLEN = GETLEN + 1
        IF (GETLEN .GT. GETMAX) THEN
            GETLEN = GETMAX
            WRITE(*,90010) 'ORAWIN_RWGETL fetch lists full'
            GOTO 30000
        ENDIF

C No leap year test -- assume 1981
    LEAP = 0
    CALL YRMIN(SRTLST(BTMPTR),MD,HN,LEAP)
    GETDAT(GETLEN) = 81000000 + MD*100 + HN/100
    GETSTN(GETLEN) = CLR
    CALL SETBIT(GETSTN(GETLEN),SRTSTN(BTMPTR)-1)
    GETMIN(GETLEN,SRTSTN(BTMPTR)) = SRTMIN(BTMPTR)

C Add station at top pointer to fetch list
25500    CALL SETBIT(GETSTN(GETLEN),SRTSTN(TOPPTR)-1)
    GETMIN(GETLEN,SRTSTN(TOPPTR)) = SRTMIN(TOPPTR)
    SPANHR = (SRTMIN(BTMPTR).GE.30) .AND. (SRTMIN(TOPPTR).LT.30)

C Move top pointer up until time period exceeded.
    TOPPTR = TOPPTR + 1
    IF (TOPPTR .LE. SRTLEN) THEN
        IF ( (SRTLST(TOPPTR) - SRTLST(BTMPTR)) .LE. TTINT) THEN
            GOTO 25500
        ENDIF
    ENDIF

C If the soundings span an hour, bump the fetch time by an hour and
C adjust the minutes
    IF (SPANHR) THEN
        YMD = GETDAT(GETLEN) / 100
        HNS = MOD(GETDAT(GETLEN),100) * 10000
        CALL TIMINC(YMD,HNS,1,3)
    
```

```

        GETDAT(GETLEN) = YMD*100 + HNS/10000
        DO 25550 J = 1, STNMAX
            IF (BITVAL(GETSTN(GETLEN),J-1) .EQ. SET) THEN
                IF (GETMIN(GETLEN,J) .GE. 30) THEN
                    GETMIN(GETLEN,J) = GETMIN(GETLEN,J) - 60
                ENDIF
            ENDIF
25550    CONTINUE
        ENDIF
        IF (TOPPTR .GT. SRTLEN) GOTO 30000

C Move bottom pointer up until time interval met
25600    BTMPTR = BTMPTR + 1
        IF ( (SRTLST(TOPPTR) - SRTLST(BTMPTR)) .LE. TTINT) THEN
            GOTO 25400
        ELSE
            GOTO 25600
        ENDIF
    ENDIF

C Output fetch lists
30000    WRITE(*,90030) GETLEN,' fetch list entries targeted : '
        FIRST = .TRUE.
30050    IF (TTINT .LT. 0) THEN
        WRITE(*,90010) ' Soundings grouped to nearest hour'
    ELSE
        WRITE(*,90015) ' Soundings grouped to within',TTINT,'minutes'
    ENDIF

    DO 30100 I = 1, GETLEN
        CALL RWSCCN(GETSTN(I),SCCTXT)
        MINMN = 60
        MINMX = -60
        DO 30075 J = 1, STNMAX
            IF (GETMIN(I,J) .NE. MSGSND) THEN
                MINMN = MIN0(MINMN,GETMIN(I,J))
                MINMX = MAX0(MINMX,GETMIN(I,J))
            ENDIF
        CONTINUE
30075    WRITE(*,90040) I,GETDAT(I),MINMN,MINMX,SCCTXT
        IF (MOD(I,2) .EQ. 0) WRITE(*,90010) ' '
30100    CONTINUE
        IF (MOD(I,2) .EQ. 0) WRITE(*,90010) ' '

C 2nd pass: list ok ?
        IF (.NOT. FIRST) THEN
            WRITE(*,90000) '0Fetch list satisfactory (Y/N) ? ',BELL
            READ (*,90010) ANS
            IF (ANS .EQ. 'Y') THEN
                GOTO 89000
            
```

```

ELSE
  WRITE(*,90000) '0Re-specify period (Y/N) ? ',BELL
  READ (*,90010) ANS
  IF (ANS .EQ. 'Y') THEN
    GOTO 10000
  ELSE
    GOTO 20000
  ENDIF
ENDIF
ELSE

```

C 1st pass : select list entries to process ?

```

  WRITE(*,90000) '0Select list entries to process (Y/N) ? ',BELL
  READ(*,90010) ANS
  IF (ANS .NE. 'Y') GOTO 89000
ENDIF

```

C Allow user to enable/disable fetch list entries

```

30200 WRITE(*,90010) '0At prompt :'
      WRITE(*,90010) '  Enter X to disable all stations'
      WRITE(*,90010) '  Enter ¼CR½ to enable all stations'
      WRITE(*,90010) '  Enter 1st char(s) of station(s) ¼no delimiter½'
      WRITE(*,90010) ' '

```

```

      K = 1
      DO 30300 I = 1, GETLEN
        WRITE(*,90045) K,GETDAT(I)
        DO 30225 J = 1, STNMAX
          IF (GETMIN(I,J) .NE. MSGSND) THEN
            WRITE(*,90046) STID01(J),GETMIN(I,J)
          ENDIF
30225  CONTINUE
        IVSCC = GETSTN(I)
        CALL RWSCCI(IVSCC,ISCC)

        IF (ISCC .GT. 0) THEN
          GETSTN(K) = ISCC
          GETDAT(K) = GETDAT(I)
          DO 30250 J = 1, STNMAX
            IF (BITVAL(GETSTN(K),J-1) .EQ. SET) THEN
              GETMIN(K,J) = GETMIN(I,J)
            ELSE
              GETMIN(K,J) = MSGSND
            ENDIF
30250  CONTINUE
          K = K + 1
        ENDIF
30300 CONTINUE
      K = K - 1

```

```

      IF (K .EQ. 0) THEN
        WRITE(*,90000) 'ONo entries selected. Re-specify period ? ',
1      BELL
        READ (*,90010) ANS
        IF (ANS .EQ. 'Y') THEN
          GOTO 10500
        ELSE
          GOTO 30000
        ENDIF
      ELSE
        GETLEN = K
        FIRST = .FALSE.
        WRITE(*,90030) GETLEN, ' Fetch list entries selected :'
        GOTO 30050
      ENDIF

*** error stop ***
88000 STOP ' RAWIN_RWGETL abnormal ending'

* Terminate RWGETL
89000 RETURN
      END

```

Fetch sounding data

```

C*****SUBROUTINE RWGETS
* * * * *
* RAWINGET_/ Get sounding data for 1 station      840715 Green *
CALLS SEARCH in UTLTYLBR.FOR to perform a directory search *
* * * * *

```

SUBROUTINE RWGETS

```
$INCLUDE: 'RAWINGET.DCL'
```

```

      INTEGER      CRDNUM,DTAFLD,FOUND,NUMCRD,BITVAL,YMD,HNS
      CHARACTER    DTAfmt*20

```

```

90000 FORMAT('RW',I6,'.',I2,A1)
90010 FORMAT('RW',I6,'.0',I1,A1)
91010 FORMAT(A2,A12)
91020 FORMAT(' RAWIN_RWGETS:',1X,A14,1X,'Unit:',I4,1X,A)
91030 FORMAT(' RAWIN_RWGETS:',1X,A14,1X,'NOT FOUND on drive')
92000 FORMAT(I6,I4,A12,10X,I8,F10.0,2X,I8,20X)
92010 FORMAT(I8,6F5.1,F4.0,38X)
92200 FORMAT(A5,I1,A14)
98000 FORMAT(' RAWIN_RWGETS--Unexpected EOF in file:',1X,A14)
98100 FORMAT(' RAWIN_RWGETS--File I/O error in file:',1X,A14)

```

*Begin RWGETS

CRDNUM = 0
SNDLEN = 0

C If sounding not flagged or sounding not available, return
IF (BITVAL(GETSTN(GETNUM),STNNUM-1) .EQ. CLR) THEN
GOTO 89900
ELSEIF (GETMIN(GETNUM,STNNUM) .EQ. MSGSND) THEN
GOTO 89900
ENDIF

C Sounding filenames are rounded to nearest hour. May have to adjust
C fetch list value (locally)

SNDGMT = GETDAT(GETNUM)
IF (GETMIN(GETNUM,STNNUM) .GE. 30) THEN
YMD = SNDGMT / 100
HNS = MOD(SNDGMT,100) * 10000
CALL TIMINC(YMD,HNS,1,3)
SNDGMT = YMD*100 + HNS/10000
ENDIF

C Build the sounding filespec.

10000 IF (MOD(SNDGMT,100) .GE. 10) THEN
WRITE(SNDNAM,90000) SNDGMT/100,MOD(SNDGMT,100),STID01(STNNUM)
ELSE
WRITE(SNDNAM,90010) SNDGMT/100,MOD(SNDGMT,100),STID01(STNNUM)
ENDIF
WRITE(SNDFIL,91010) DRIVE(SNDDX), SNDNAM

C Open sounding file. Error return if not disk-resident.

CALL SEARCH (SNDNAM,SNDDX,FOUND)
IF (FOUND .EQ. 0) THEN
OPEN(SNDLU,FILE=SNDFIL,STATUS='OLD')
WRITE(*,91020) SNDFIL, SNDLU, 'Opened'
ELSE
WRITE(*,91030) SNDFIL
SNDLEN = -1
GOTO 89900
ENDIF

C Read in the header information

20000 READ(SNDLU,92000,END=88000,ERR=88100) SNDMDT,SNDMIN,SNDSTN,SNDRPT,
SNDHGT,SNDLDP
READ(SNDLU,92010,END=88000,ERR=88100) SNDGMT,SNDLAT,SNDLON,SNDMSG

C Read in data to requested pressure level

21000 NUMCRD = (SNDRPT - 1) / 3 + 1
22000 CRDNUM = CRDNUM + 1
IF (CRDNUM .LT. NUMCRD) THEN

```

      DTAFLD = 3
    ELSEIF (CRDNUM .EQ. NUMCRD) THEN
      DTAFLD = MOD(SNDRPT+3,3)
      IF (DTAFLD .EQ. 0) DTAFLD = 3
    ELSEIF (CRDNUM .GT. NUMCRD) THEN
      GOTO 89000
    ENDIF
    SNDLEN = SNDLEN + DTAFLD
    IF (SNDLEN .GT. SNDMAX) THEN
      DTAFLD = SNDLEN - SNDMAX
      SNDLEN = SNDMAX
    ENDIF

    WRITE(DTAFMT,92200) ' (5X,', DTAFLD, '(5F4.1,F5.2)) '
    READ(SNDLU,DTAFMT,END=88000,ERR=88100)
    - ((SNDSTA(I,J),J=1,SNDPRM),I=SNDLEN-DTAFLD+1,SNDLEN)
    IF((SNDLEN .LT. SNDMAX) .AND. (SNDSTA(SNDLEN,1) .GT. SNDPRL)) THEN
      GOTO 22000
    ELSE
      GOTO 89000
    ENDIF

*** unexpected EOF ***
88000 WRITE(*,98000) SNDFIL
      SNDLEN = -2
      GOTO 89000

*** sounding file read error ***
88100 WRITE(*,98100) SNDFIL
      SNDLEN = -3

C Close sounding file
89000 CLOSE(SNDLU)
      WRITE(*,91020) SNDFIL,SNDLU,'Closed'

*Terminate RWGETS
89900 RETURN
      END

```

Access index file

```

C*****SUBROUTINE RWGETX
* * * * *
* 15 July 1984 Green
*   Access index file to find soundings available for current
*   period.
*
*Subprogram references
CALLS SEARCH in UTLTYLBR.FOR to do a directory search

```

SUBROUTINE RWGETX

\$INCLUDE: 'RAWINGET.DCL'

INTEGER FOUND, MINUTE(6), TIMGMT

```
90000 FORMAT(A)
91000 FORMAT(A2,A12)
91010 FORMAT(' RAWIN_RWGETX:',1X,A14,1X,'Unit:',I4,1X,A)
91020 FORMAT(' RAWIN_RWGETX:',1X,A14,1X,'NOT FOUND on drive')
92000 FORMAT(I8,6(3X,I3))
98000 FORMAT(' RAWIN_RWGETX--file I/O error in ',A14)
```

*Begin RWGETX

IDXLEN = 0

C Build the index file spec and open it. Error return if not found

```
10000 WRITE(IDXFIL,91000) DRIVE(IDXDX), IDXNAM
      CALL SEARCH (IDXNAM,IDXDX,FOUND)
      IF (FOUND .EQ. 0) THEN
        OPEN(IDXLU,FILE=IDXFIL,STATUS='OLD')
        WRITE(*,91010) IDXFIL, IDXLU, 'Opened'
      ELSE
        IDXLEN = -1
        WRITE(*,91020) IDXFIL
        GOTO 89900
      ENDIF
```

C Get soundings available for current period

```
20000 READ(IDXLU,92000,END=89000,ERR=88000)TIMGMT,(MINUTE(J),J=1,STNMAX)
      IF ( (TIMGMT .GE. BEGDAT) .AND. (TIMGMT .LE. ENDDAT) ) THEN
        IDXLEN = IDXLEN + 1
        IF (IDXNUM .GT. IDXMAX) THEN
          IDXLEN = IDXMAX
          WRITE(*,90000) 'ORAWIN_RWGETX index list full'
          WRITE(*,90000) ' Other entries exist for requested period'
          GOTO 89000
        ENDIF
```

```
      IDXDAT(IDXLEN) = TIMGMT
      DO 21000 STNNUM = 1, STNMAX
        IDXMIN(IDXLEN,STNNUM) = MINUTE(STNNUM)
```

21000 CONTINUE

```
      ELSEIF (TIMGMT .GT. ENDDAT) THEN
        GOTO 89000
      ENDIF
```

GOTO 20000

*** index file read error ***

88000 WRITE(*,98000) IDXFIL

IDXLEN = -2

C Close index file

89000 CLOSE(IDXLU)

WRITE(*,91010) IDXFIL, IDXLU, 'Closed'

*Terminate RWGETX

89900 RETURN

END

Fetch module common initialization

C*****SUBROUTINE RWGETZ.

```

* * * * *
* 15 July 1984  Green      RAWINGET subprogram
*
*      RAWINGET common initialization
*      /RWMSC/ /RWMSCC/ /RWIDX/ /RWIDXC/ /RWSTN/ /RWSTNC/
*      /RWSND/ /RWSNDC/
*
* Logical unit assignments:
*      IDXLU = 80  index file LU -- not opened locally
*      SNDLU = 81  sounding file LU -- not opened locally
*
* Subprogram references:
CALLS DFDRV      in DFDRV.ASM to get default drive number
*
* Other external datasets required:
*      RWMSC.DCL  -- RAWIN miscellaneous common block
*      RWSTN.DCL  -- RAWIN station data common block
*      RWIDX.DCL  -- RAWIN index file common block
*      RWSND.DCL  -- RAWIN sounding file common block
*
* Notes:
* (1) RWGETZ should be called prior to calling any of the fetch
*      routines in the RAWINGET module.
* (2) The index and sounding file drive numbers (IDXDX,SNDDX) are
*      set to the default drive. The caller should overwrite these
*      if these datafiles are not on the default drive.
* * * * *

```

SUBROUTINE RWGETZ

```
$INCLUDE:'RWMSC.DCL'  
$INCLUDE:'RWSTN.DCL'  
$INCLUDE:'RWIDX.DCL'  
$INCLUDE:'RWSND.DCL'
```

*Begin RWGETZ

C Miscellaneous information initialization

```
CLR = 0  
SET = 1  
BIT0 = 1  
BIT1 = 2  
BIT2 = 4  
BIT3 = 8  
BIT4 = 16  
BIT5 = 32  
BIT6 = 64  
BIT7 = 128
```

```
DRIVE(1) = 'A:'  
DRIVE(2) = 'B:'  
DRIVE(3) = 'C:'  
DRIVE(4) = 'D:'  
DRIVE(5) = 'E:'  
DRIVE(6) = 'F:'  
DRIVE(7) = 'G:'  
DRIVE(8) = 'H:'  
DRIVE(9) = 'I:'
```

```
MONTH(1) = 'Jan'  
MONTH(2) = 'Feb'  
MONTH(3) = 'Mar'  
MONTH(4) = 'Apr'  
MONTH(5) = 'May'  
MONTH(6) = 'Jun'  
MONTH(7) = 'Jul'  
MONTH(8) = 'Aug'  
MONTH(9) = 'Sep'  
MONTH(10) = 'Oct'  
MONTH(11) = 'Nov'  
MONTH(12) = 'Dec'
```

C Index file data area initialization

```
IDXLU = 80  
CALL DFDRV(I)  
IDXDX = I  
IDXMAX = 25  
IDXNAM = 'RWNSONDE.IDX'  
MSGSD = 999  
GETMAX = 50
```

C Station data area initialization

STNMAX = 6

C Station locations:

C Baker

STLATX(1,1) = 46.0
STLATX(2,1) = 21.0
STLATX(3,1) = 10.6
STLONX(1,1) = 104.0
STLONX(2,1) = 13.0
STLONX(3,1) = 26.3

C Colstrip

STLATX(1,2) = 45.0
STLATX(2,2) = 53.0
STLATX(3,2) = 14.1
STLONX(1,2) = 106.0
STLONX(2,2) = 37.0
STLONX(3,2) = 47.5

C Glendive

STLATX(1,3) = 47.0
STLATX(2,3) = 5.0
STLATX(3,3) = 0.0
STLONX(1,3) = 105.0
STLONX(2,3) = 6.0
STLONX(3,3) = 42.6

C Knowlton

STLATX(1,4) = 46.0
STLATX(2,4) = 20.0
STLATX(3,4) = 13.8
STLONX(1,4) = 105.0
STLONX(2,4) = 4.0
STLONX(3,4) = 40.0

C Miles City

STLATX(1,5) = 46.0
STLATX(2,5) = 26.0
STLATX(3,5) = 5.5
STLONX(1,5) = 105.0
STLONX(2,5) = 52.0
STLONX(3,5) = 30.0

C Powderville

STLATX(1,6) = 45.0
STLATX(2,6) = 45.0
STLATX(3,6) = 20.3

```
STLONX(1,6) = 105.0
STLONX(2,6) = 6.0
STLONX(3,6) = 7.0
```

C Station Heights

```
STHGTX(1) = 903.
STHGTX(2) = 1053.
STHGTX(3) = 803.
STHGTX(4) = 954.
STHGTX(5) = 800.
STHGTX(6) = 866.
```

C Station id codes

```
STID12(1) = 'BAKER      '
STID12(2) = 'COLSTRIP   '
STID12(3) = 'GLEN DIVE  '
STID12(4) = 'KNOWLTON   '
STID12(5) = 'MILES CITY '
STID12(6) = 'POWDERVILLE'
```

```
STID04(1) = 'BAK '
STID04(2) = 'COL '
STID04(3) = 'GDV '
STID04(4) = 'KWL '
STID04(5) = 'MLS '
STID04(6) = 'POW '
```

```
STID01(1) = 'B'
STID01(2) = 'C'
STID01(3) = 'G'
STID01(4) = 'K'
STID01(5) = 'M'
STID01(6) = 'P'
```

C Sounding file data area initialization

```
SNDLU = 81
CALL DFDRV(I)
SNDDX = 1
SNDPRM = 6
SNDMAX = 200
MDTGMT = 6
SNDMSG = 999.0
```

*Terminate RWGETZ

```
RETURN
END
```

RAWIN jobfile access

```

C*****SUBROUTINE RWJOB(NUMTSK,TSKNUM,GET,IWRIT)
* * * * *
* RAWIN jobfile write/read                                840820 Green *
*   NUMTSK [I]   number of tasks                          *
*   TSKNUM [I]   current task number                      *
*   GET      [I]   current fetch list number              *
*   IWRIT    [I]   read/write flag :                      *
*                   =0 write      1 read                  *
* Jobfile opened/closed locally on unit 84                *
* * * * *

```

```

SUBROUTINE RWJOB(NUMTSK,TSKNUM,GET,IWRIT)

```

```

$INCLUDE: 'RAWINTSK.DCL'

```

```

INTEGER    NUMTSK,TSKNUM,GET,IWRIT,JOBLU
CHARACTER  JOBFIL*14

```

```

DATA JOBLU/84/

```

```

90020 FORMAT(A2,A12)
90030 FORMAT(' RAWIN_RWJOB:',1X,A14,1X,'Unit:',1X,I2,1X,A)
90040 FORMAT(1X,4I10)

```

```

*Begin RWJOB

```

```

WRITE(JOBFIL,90020) DRIVE(TSKDX),'RAWINJOB.DTA'
IF (IWRIT .EQ. 0) THEN
  OPEN(JOBLU,FILE=JOBFIL,STATUS='NEW')
  WRITE(*,90030) JOBFIL,JOBLU,'Opened'
  WRITE(JOBLU,90040) NUMTSK,TSKNUM,GET,GETLEN
ELSE
  OPEN(JOBLU,FILE=JOBFIL,STATUS='OLD')
  WRITE(*,90030) JOBFIL,JOBLU,'Opened'
  READ(JOBLU,90040) NUMTSK,TSKNUM,GET,GETLEN
ENDIF
CLOSE(JOBLU)
WRITE(*,90030) JOBFIL,JOBLU,'Closed'

```

```

*Terminate RWJOB
89000 RETURN
END

```

RAWIN job processing initialization

C*****SUBROUTINE RWJOBZ

* * * * *

* RAWINJ initialization 840820 Green *

CALLS RWTSKZ *

CALLS DFDRV *

* * * * *

SUBROUTINE RWJOBZ

\$INCLUDE: 'RAWINJOB.DCL'

REAL RMN,RMX

DATA RMN,RMX /-3.0E-39,1.7E+38/

*Begin RWJOBZ

C Initialize frontend parms

CALL RWTSKZ

C Table initialization

TBLU = 83

CALL DFDRV(I)

TBLDX = I

TBLMAX = 400

TBLNAM = 'LPT1:'

TBLFIL = 'LPT1:'

C Map initialization

C Set reference and fiducial points to PMKIN values

LATR = 46.4281

LONR = 105.8733

LATF = 45.75

LONF = 105.90

C Convert station locations to decimal format

DO 10 I = 1, STNMAX

LAT(I) = STLATX(1,I) + STLATX(2,I)/60. + STLATX(3,I)/3600.

LON(I) = STLONX(1,I) + STLONX(2,I)/60. + STLONX(3,I)/3600.

10 CONTINUE

C Find cartesian distances of fiducial point from projected north pole.

CALL LAMB(LATR,LONR,LATF,LONF,DUM,DUM,XF,YF,DUM,DUM,.TRUE.)

C Find cartesian distances of stations from fiducial point.

DO 20 I = 1, STNMAX

```

      CALL LAMB(LATR,LONR,LATF,LONF,LAT(I),LON(I),XF,YF,XS(I),YS(I),
-          .FALSE.)
20    CONTINUE

C Assume no balloon drift for now (This'll have to be moved elsewhere
C   when BD implemented)
      DO 30 J = 1, STNMAX
      DO 30 I = 1, SNDMAX
          XDX(I,J) = 0.0
          YDX(I,J) = 0.0
30    CONTINUE

*Terminate RWJOBZ
      RETURN
      END

```

Calculate MSL heights

```

C*****SUBROUTINE RWMSLC
* * * * *
* Calculate MSL heights [m]                                840820 Green *
CALLS BITVAL                                              *
CALLS DZGPM                                              *
* * * * *

```

SUBROUTINE RWMSLC

\$INCLUDE: 'RAWINJOB.DCL'

```

      LOGICAL HT
      INTEGER BITVAL,L1,L2
      REAL DH

```

*Begin RWMSLC

```

      DO 10 J = 1, STNMAX
      DO 10 I = 1, SNDMAX
          HMX(I,J) = SNDMSG
10    CONTINUE

C See if a height parameter is required
      HT = .FALSE.
      IF (TSNUM.EQ. MP) THEN
          DO 10000 TBCNUM = 1, TBCLEN
              IF(.NOT. HT) THEN
                  HT = (TBCSPC(TBCNUM).GE.2) .AND. (TBCSPC(TBCNUM).LE.5)
              ENDIF
10000    CONTINUE
          ELSE

```

```

DO 15000 TBHNUM = 1, TBHLEN
  IF(.NOT. HT) THEN
    HT = (TBHSPC(TBHNUM).GE.2) .AND. (TBHSPC(TBHNUM).LE.5)
  ENDIF
15000 CONTINUE
ENDIF
IF(.NOT. HT) HT = (TBSPC.GE.2) .AND. (TBSPC.LE.5)

C If a height parameter is required, then calculate MSL heights [m]
IF (HT) THEN
DO 18000 STNNUM = 1, STNMAX
  IF (BITVAL(STNREQ,STNNUM-1) .EQ. SET) THEN
    HMX(1,STNNUM) = STHGTX(STNNUM)
    L1 = 1
    DO 17000 L2 = 2, NX(STNNUM)
      DH = DZGPM(TEX(L1,STNNUM),TDX(L1,STNNUM),PRX(L1,STNNUM),
-          TEX(L2,STNNUM),TDX(L2,STNNUM),PRX(L2,STNNUM),
-          SNDMSG)
      IF (DH .NE. SNDMSG) THEN
        L1 = L1 + 1
        HMX(L1,STNNUM) = HMX(L1-1,STNNUM) + DH
      ENDIF
17000 CONTINUE
      NX(STNNUM) = L1
    ENDIF
18000 CONTINUE
  ENDIF
ENDIF

*Terminate RWMSLC
89000 RETURN
END

```

Table interpolation

```

C*****SUBROUTINE RWNTRP
* * * * *
* Interpolate for values at current RAWIN table ordinate 840820 Green *
CALLS BITVAL
CALLS UUVV
CALLS XINTRP
CALLS WIND
* * * * *

```

SUBROUTINE RWNTRP

\$INCLUDE: 'RAWINJOB.DCL'

INTEGER NPD,BITVAL

DATA NPD /11/

*Begin RWNTRP

C If not first table entry, save previous interpolated values

10000 IF (TBLLIN .GT. 1) THEN

DO 11000 I = 1, STNMAX

PRI0(I) = PRI(I)

TEI0(I) = TEI(I)

TDI0(I) = TDI(I)

WSI0(I) = WSI(I)

WDI0(I) = WDI(I)

TMI0(I) = TMI(I)

HMI0(I) = HMI(I)

UUI0(I) = UUI(I)

VVI0(I) = VVI(I)

XDI0(I) = XDI(I)

YDI0(I) = YDI(I)

11000 CONTINUE

ENDIF

C Assign missing values to current data

DO 12000 I = 1, NPD * STNMAX

PDI(I) = SNDMSG

12000 CONTINUE

C Find lower and upper primary data boundaries

DO 15000 STNNUM = 1, STNMAX

IF (BITVAL(STNREQ,STNNUM-1) .EQ. SET) THEN

DO 13000 I = 1, NX(STNNUM)

IF (YYX(I,STNNUM) .NE. SNDMSG) THEN

IF (TBYSPEC .EQ. PR) THEN

IF (YYX(I,STNNUM) .LE. TBLVAL(TBLLIN,1)) GOTO 14000

ELSE

IF (YYX(I,STNNUM) .GE. TBLVAL(TBLLIN,1)) GOTO 14000

ENDIF

ENDIF

13000 CONTINUE

GOTO 15000

14000 J = I - 1

C Ordinate value below surface ? Return surface values

IF (J .EQ. 0) THEN

PRI(STNNUM) = PRX(1,STNNUM)

TEI(STNNUM) = TEX(1,STNNUM)

TDI(STNNUM) = TDX(1,STNNUM)

WSI(STNNUM) = WSX(1,STNNUM)

WDI(STNNUM) = WDX(1,STNNUM)

TMI(STNNUM) = TMX(1,STNNUM)

```

HMI(STNNUM) = HMX(1,STNNUM)
CALL UUVV(WSI(STNNUM),WDI(STNNUM),SNDMSG,
          UI(STNNUM),VVI(STNNUM))
XDI(STNNUM) = 0.0
YDI(STNNUM) = 0.0

```

C No ? Then interpolate to find values. Use log interpolation for PR
ELSE

```

IF (TBYSPEC.EQ. PR) THEN
  Y = ALOG(TBLVAL(TBLLIN,1))
  Y1 = ALOG(YYX(J,STNNUM))
  Y2 = ALOG(YYX(I,STNNUM))
ELSE
  Y = TBLVAL(TBLLIN,1)
  Y1 = YYX(J,STNNUM)
  Y2 = YYX(I,STNNUM)
ENDIF
X1 = ALOG(PRX(J,STNNUM))
X2 = ALOG(PRX(I,STNNUM))
PRI(STNNUM) = XINTRP(Y1,Y2,Y,X1,X2,SNDMSG)
PRI(STNNUM) = EXP(PRI(STNNUM))
X1 = TEX(J,STNNUM)
X2 = TEX(I,STNNUM)
TEI(STNNUM) = XINTRP(Y1,Y2,Y,X1,X2,SNDMSG)
X1 = TDX(J,STNNUM)
X2 = TDX(I,STNNUM)
TDI(STNNUM) = XINTRP(Y1,Y2,Y,X1,X2,SNDMSG)
X1 = TMX(J,STNNUM)
X2 = TMX(I,STNNUM)
TMI(STNNUM) = XINTRP(Y1,Y2,Y,X1,X2,SNDMSG)
X1 = HMX(J,STNNUM)
X2 = HMX(I,STNNUM)
HMI(STNNUM) = XINTRP(Y1,Y2,Y,X1,X2,SNDMSG)
X1 = WSX(J,STNNUM)
X2 = WSX(I,STNNUM)
Z1 = WDX(J,STNNUM)
Z2 = WDX(I,STNNUM)
CALL UUVV(X1,Z1,SNDMSG,U1,V1)
CALL UUVV(X2,Z2,SNDMSG,U2,V2)
UI(STNNUM) = XINTRP(Y1,Y2,Y,U1,U2,SNDMSG)
VVI(STNNUM) = XINTRP(Y1,Y2,Y,V1,V2,SNDMSG)
CALL WIND(UI(STNNUM),VVI(STNNUM),SNDMSG,
          WDI(STNNUM),WSI(STNNUM))
XDI(STNNUM) = 0.0
YDI(STNNUM) = 0.0

```

ENDIF

ENDIF

15000 CONTINUE

*Terminate RWNTRP

89000 RETURN
END

Input operation code

```
C*****SUBROUTINE RWOPCI (NOPCOD,OPCOD,OPTXT,OPFLG,OPTTL,OPBIT,OPNUM)
* * * * *
* Input RAWIN operation code                                840820 Green *
*   NOPCOD [I]      number of operation codes                *
*   OPCOD [C*2]     array (NOPCOD) containing operation codes *
*   OPTXT [C*30]    array (NOPCOD) containing opcode descriptions *
*   OPFLG [I]       array (NOPCOD) containing opcode flags    *
*   OPTTL [C*30]    operation code title                      *
*   OPBIT [I]       bits in OPFLG to test                     *
*   OPNUM [I]       operation code number, or zero if exit    *
CALLS BITVAL
* (1) The opcode EX (exit) is enabled locally                *
* * * * *
```

SUBROUTINE RWOPCI (NOPCOD,OPCOD,OPTXT,OPFLG,OPTTL,OPBIT,OPNUM)

\$INCLUDE: 'RAWINTSK.DCL'

```
INTEGER      NOPCOD,OPFLG(1),OPBIT,OPNUM,BITVAL
CHARACTER    OPCOD(1)*2,OPTXT(1)*30,OPTTL*30,ENTRY*2,EXIT*2
CHARACTER    BELL*1
LOGICAL      VALID
```

DATA EXIT /'EX'/

*Begin RWOPCI

BELL = CHAR(7)

C Input operation code

```
10000 WRITE(*,'(/,1X,5HInput,1X,A30,1X,1H:,1X,A,*)') OPTTL,BELL
      READ (*,'(A2)') ENTRY
```

C Test entry for validity

DO 10100 I = 1, NOPCOD

C note: all test bits must be enabled

DO 10050 J = 1, 8

IF (BITVAL(OPBIT,J-1) .EQ. SET) THEN

VALID = BITVAL(OPFLG(I),J-1) .EQ. CLR

IF (.NOT. VALID) GOTO 10100

ENDIF

10050 CONTINUE

```

C operation code enabled. test entry
      IF (ENTRY .EQ. OPCOD(I)) GOTO 11000
10100 CONTINUE

C entry not on operation code list. test for exit
      IF (ENTRY .EQ. EXIT) THEN
        I = 0
        GOTO 11000

C invalid entry -- output valid operation codes
      ELSE
        WRITE(*,'(A)') '0Valid codes are :'
        K = 0
        DO 10200 I = 1, NOPCOD

C all test bits must be enabled
          DO 10150 J = 1, 8
            IF (BITVAL(OPBIT,J-1) .EQ. SET) THEN
              VALID = BITVAL(OPFLG(I),J-1) .EQ. CLR
              IF (.NOT. VALID) GOTO 10200
            ENDIF
10150    CONTINUE
          K = K + 1
          WRITE(*,'(3X,A2,2X,A30,*)') OPCOD(I),OPTXT(I)
          IF (MOD(K+2,2) .EQ. 0) WRITE(*,'(1X)')
10200    CONTINUE
          WRITE(*,'(A)') '    EX  exit'
          GOTO 10000
        ENDIF
11000  OPNUM = I
        IF (OPNUM .NE. 0) THEN
          WRITE(*,'(8X,A30,1X,A2,1X,A30)') OPTTL,OPCOD(OPNUM),OPTXT(OPNUM)
        ENDIF

*Terminate RWOPCI
89000 RETURN
      END

```

Set plot specifications

```

C*****SUBROUTINE RWPLTI
*****
* Set RAWIN plot specifications                                840820 Green *
CALLS RWSCCN                                                    *
CALLS SETBIT                                                    *
*****

```

```

SUBROUTINE RWPLTI

```

```
$INCLUDE: 'RAWINTSK.DCL'
```

```
CHARACTER SCCTXT(12)*1
```

```
90000 FORMAT(A)
```

```
90010 FORMAT(1X,12A1,1X,': ',1X,A2,1X,'vs',1X,A2,2X,'?',1X,*)
```

```
*Begin RWPLTI
```

```
10000 WRITE(*,90000) 'OAt prompt, enter ¼CR½ to enable, X to disable'
```

```
DO 10100 I = 1, TBHLEN
```

```
IF (TSNUM .EQ. MP) CALL RWSCCN(TBHSPC(I),SCCTXT)
```

```
PFCSPC(I) = CLR
```

```
DO 10050 K = 1, TBCLEN
```

```
IF (TSNUM .EQ. MP) THEN
```

```
WRITE(*,90010) (SCCTXT(J),J=1,12),PMOPC(TBYSPC),  
PMOPC(TBCSPC(K))
```

```
ELSEIF (TSNUM .EQ. IP) THEN
```

```
CALL RWSCCN(TBCSPC(K),SCCTXT)  
WRITE(*,90010) (SCCTXT(J),J=1,12),PMOPC(TBYSPC),  
PMOPC(TBHSPC(I))
```

```
ENDIF
```

```
READ(*,90000) CHR
```

```
IF (CHR .EQ. ' ') CALL SETBIT(PFCSPC(I),K-1)
```

```
10050 CONTINUE
```

```
10100 CONTINUE
```

```
*Terminate RWPLTI
```

```
89000 RETURN
```

```
END
```

Calculate abscissa parameter

```
C*****SUBROUTINE RWPMXC(SCC,PMNUM,PMVAL)
```

```
*****
```

```
* Calculate abscissa parameter for RAWIN table 840820 Green *
```

```
CALLS RWAVG *
```

```
CALLS WIND *
```

```
CALLS TVIRT *
```

```
CALLS TPOT *
```

```
CALLS TEQPOT *
```

```
CALLS VAPOR *
```

```
CALLS MIXRAT *
```

```
CALLS SPFHUM *
```

```
CALLS RELHUM *
```

```
CALLS PCON *
```

```
CALLS PVORT *
```

```
CALLS PMCON *
```

SUBROUTINE RWPMXC(SCC,PMNUM,PMVAL)

\$INCLUDE: 'RAWINJOB.DCL'

```

INTEGER    SCC,PMNUM,NSCC,BITVAL
REAL       PMVAL,HX(6),FEET
REAL       XXA(6),YYA(6),UUA(6),VVA(6),PRA(6),TEA(6),RHA(6)
REAL       HMA(6),HMAO(6),XXAO(6),YYAO(6),UUAO(6),VVAO(6)
REAL       RWAVG,TVIRT,TPOT,TEQPOT,VAPOR,MIXRAT,SPFHUM,RELHUM

```

DATA FEET /3.28084/

*Begin RWPMXC

C Find average values for the primary data

```

10000 P = RWAVG(SCC,PRI,STNMAX,SNDSMSG)
      T = RWAVG(SCC,TEI,STNMAX,SNDSMSG)
      D = RWAVG(SCC,TDI,STNMAX,SNDSMSG)
      C = RWAVG(SCC,TMI,STNMAX,SNDSMSG)
      H = RWAVG(SCC,HMI,STNMAX,SNDSMSG)
      U = RWAVG(SCC,UUI,STNMAX,SNDSMSG)
      V = RWAVG(SCC,VVI,STNMAX,SNDSMSG)
      CALL WIND(U,V,SNDSMSG,W,S)

```

C Build areal parameter arrays

```

      IF ( (PMNUM .EQ. CV) .OR. (PMNUM .EQ. RV) .OR.
-        (PMNUM .EQ. AV) .OR. (PMNUM .EQ. MC) .OR.
-        (PMNUM .EQ. WW) ) THEN
        NSCC = 0
        DO 10010 I = 1, STNMAX
          IF (BITVAL(SCC,I-1) .EQ. SET) THEN
            NSCC = NSCC + 1
            HMA(NSCC) = HMI(I)
            XXA(NSCC) = XS(I) + XDI(I)
            YYA(NSCC) = YS(I) + YDI(I)
            UUA(NSCC) = UUI(I)
            VVA(NSCC) = VVI(I)
            PRA(NSCC) = PRI(I)
            TEA(NSCC) = TEA(I)
            RHA(NSCC) = RELHUM(TEI(I),TDI(I),PRI(I),SNDSMSG) / 100.
            IF (TBLLIN .GT. 1) THEN
              HMAO(NSCC) = HMIO(I)
              XXAO(NSCC) = XS(I) + XDIO(I)
              YYAO(NSCC) = YS(I) + YDIO(I)
              UUAO(NSCC) = UUIO(I)
              VVAO(NSCC) = VVIO(I)
            ENDIF
          ENDIF
        END DO

```

```

      ENDIF
10010  CONTINUE
      ENDIF

```

C Jump to calculate parameter value

```

      GOTO (20000,20100,20200,20300,20400,20500,20600,20700,20800,20900,
-       21000,21100,21200,21300,21400,21500,21600,21700,21800,21900,
-       22000,22100,22200,22300,22400,22500,22600,22700,22800,22900)
-       PMNUM

```

C PR pressure [mb]
20000 PMVAL = P
 GOTO 89000

C HM height MSL [m]
20100 PMVAL = H
 GOTO 89000

C ZM height AGL [m]
20200 DO 20210 STNNUM = 1, STNMAX
 IF (HMI(STNNUM) .NE. SNDMSG) THEN
 HX(STNNUM) = HMI(STNNUM) - STHGTX(STNNUM)
 ELSE
 HX(STNNUM) = SNDMSG
 ENDIF
20210 CONTINUE
 PMVAL = RWAVG(SCC,HX,STNMAX,SNDMSG)
 GOTO 89000

C HF height MSL [ft]
20300 DO 20310 STNNUM = 1, STNMAX
 IF (HMI(STNNUM) .NE. SNDMSG) THEN
 HX(STNNUM) = HMI(STNNUM) * FEET
 ELSE
 HX(STNNUM) = SNDMSG
 ENDIF
20310 CONTINUE
 PMVAL = RWAVG(SCC,HX,STNMAX,SNDMSG)
 GOTO 89000

C ZF height AGL [ft]
20400 DO 20410 STNNUM = 1, STNMAX
 IF (HMI(STNNUM) .NE. SNDMSG) THEN
 HX(STNNUM) = (HMI(STNNUM) - STHGTX(STNNUM)) * FEET
 ELSE
 HX(STNNUM) = SNDMSG
 ENDIF
20410 CONTINUE
 PMVAL = RWAVG(SCC,HX,STNMAX,SNDMSG)
 GOTO 89000

C TM time after release [min]

20500 PMVAL = C
GOTO 89000

C TE temperature [C]

20600 PMVAL = T
GOTO 89000

C TD dewpoint [C]

20700 PMVAL = D
GOTO 89000

C TV virtual temperature [K]

20800 PMVAL = TVIRT(T,D,P,SNDSMSG)
GOTO 89000

C TP potential temperature [K]

20900 PMVAL = TPOT(T,P,SNDSMSG)
GOTO 89000

C TQ equivalent potential temperature [K]

21000 PMVAL = TEQPOT(T,D,P,SNDSMSG)
GOTO 89000

C WS wind speed [m/s]

21100 PMVAL = S
GOTO 89000

C WD wind direction [deg]

21200 PMVAL = W
GOTO 89000

C UU eastward wind component [m/s]

21300 PMVAL = U
GOTO 89000

C VV northward wind component [m/s]

21400 PMVAL = V
GOTO 89000

C EA vapor pressure [mb]

21500 PMVAL = VAPOR(D,SNDSMSG)
GOTO 89000

C ES saturation vapor pressure [mb]

21600 PMVAL = VAPOR(T,SNDSMSG)
GOTO 89000

C MR mixing ratio [g/g]

21700 PMVAL = MIXRAT(D,P,SNDSMSG)
GOTO 89000

C SH specific humidity [g/g]
21800 PMVAL = SPFHUM(D,P,SNDSMSG)
GOTO 89000

C RH relative humidity [%]
21900 PMVAL = RELHUM(T,D,P,SNDSMSG)
GOTO 89000

C CV convergence [1/s]
22000 CALL PCON(NSCC,XXA,YYA,UUA,VVA,SNDSMSG,PMVAL)
GOTO 89000

C RV relative vorticity [1/s]
22100 CALL PVORT(NSCC,XXA,YYA,UUA,VVA,SNDSMSG,PMVAL)
GOTO 89000

C AV absolute vorticity [1/s]
22200 PMVAL = -1.
GOTO 89000

C MC moisture convergence [g/m³-s]
22300 CALL PMCON(NSCC,XXA,YYA,PRA,TEA,RHA,UUA,VVA,SNDSMSG,PMVAL)
GOTO 89000

C AE atmospheric energy [cal/g]
22400 R = RELHUM(T,D,P,SNDSMSG)/100.
CALL TOTAL(U,V,T,R,H,P,SNDSMSG,PMVAL)
GOTO 89000

C PE potential energy
22500 PMVAL = -1.
GOTO 89000

C KE kinetic energy
22600 PMVAL = -1.
GOTO 89000

C XD eastward drift component [m]
22700 PMVAL = X
GOTO 89000

C YD northward drift component [m]
22800 PMVAL = Y
GOTO 89000

C WW vertical wind speed [m/s]
22900 IF (TBLLIN.EQ.1) THEN

```

        PMVAL = 0.0
    ELSE
        IF (TBLVAL(TBLLIN-1,TBCNUM+1) .EQ. SNDMSG) THEN
            PMVAL = SNDMSG
        ELSE
            CALL PCON(NSCC,XXA0,YYA0,UUA0,VVA0,SNDMSG,C0)
            CALL PCON(NSCC,XXA,YYA,UUA,VVA,SNDMSG,C1)
            IF ( (C0 .EQ. SNDMSG) .OR. (C1 .EQ. SNDMSG) ) THEN
                PMVAL = SNDMSG
            ELSE
                C = (C0 + C1) / 2.
                W = 0.0
                DO 22910 I = 1, NSCC
                    DH = HMA(I) - HMA0(I)
                    W = W + DH * C
22910          CONTINUE
                PMVAL = W/NSCC + TBLVAL(TBLLIN-1,TBCNUM+1)
            ENDIF
        ENDIF
    ENDIF
    GOTO 89000

*Terminate RWPMXC
89000 RETURN
END

```

Input abscissa parameter

```

C*****SUBROUTINE RWPMXI (YPM,MXXPM,XPM,NXPM)
* * * * *
* Input RAWIN abscissa parameters                                     840820 Green *
*      YPM [I]      ordinate parameter number for table             *
*      MXXPM [I]     maximum number of abscissa parameters          *
*      XPM [I]       array (MXXPM) containing abscissa parm nos.    *
*      NXPM [I]      number of abscissa parameters entered.         *
CALLS RWOPCI
* (1) The areal parameter flag must be set to permit parameters    *
*      requiring 3 or more stations to be enabled.                  *
* (2) The ordinate parameter is not allowed as an abscissa parameter.*
* (3) If YPM is zero on entry, NXPM is returned zero.              *
* * * * *

```

SUBROUTINE RWPMXI (YPM,MXXPM,XPM,NXPM)

\$INCLUDE: 'RAWINTSK.DCL'

INTEGER YPM,MXXPM,XPM(MXXPM),NXPM,TSTBIT

*Begin RWPMXI

C If undefined ordinate parameter, exit.

```

      IF (YPM .EQ. 0) THEN
        NXPM = 0
        GOTO 89000
      ENDIF

```

C Set test bits

```

      IF (APFLG) THEN
        TSTBIT = BIT0
      ELSE
        TSTBIT = BIT0 + BIT2
      ENDIF

```

C Input the abscissa parameters

```

      WRITE(*,'(A)') '0Abscissa parameter specification'
10000 DO 10100 I = 1, MXXPM
10005   WRITE(*,'(A,I2,*)') ' X Parameter # ',I
        CALL RWOPCI(PMLEN,PMOPC,PMDES,PMBIT,PMTTL,TSTBIT,XPM(I))
        IF (XPM(I) .EQ. 0) THEN
          GOTO 10200
        ELSE IF (XPM(I) .EQ. YPM) THEN
          WRITE(*,'(A)') '0Ordinate parm invalid for abscissa parm'
          GOTO 10005
        ENDIF
10100 CONTINUE
10200 NXPM = I - 1

```

*Terminate RWPMXI

89000 RETURN

END

Calculate ordinate value

C*****SUBROUTINE RWPMYC(SCC)

```

*****
* Calculate ordinate value for RAWIN table                                840820 Green *
CALLS BITVAL                                                                *
CALLS BSORT                                                                *
*****

```

SUBROUTINE RWPMYC(SCC)

\$INCLUDE: 'RAWINJOB.DCL'

```

      INTEGER SCC,BITVAL
      REAL RMN,RMX

```

DATA RMN,RMX /-3.0E-39,1.7E+38/

90000 FORMAT(A)

*Begin RWPMYC

```

      DO 10 J = 1, TBCMAX + 1
      DO 10 I = 1, TBLMAX
        TBLVAL(I,J) = SNDMSG
10    CONTINUE

C Contact point mode ? Transfer data directly to tables
      IF (TBYINC .EQ. 0.0) THEN
        DO 10000 STNNUM = 1, STNMAX
          IF (BITVAL(SCC,STNNUM-1) .EQ. SET) GOTO 11000
10000  CONTINUE
        WRITE(*,90000) 'ORAWINJ_RWPMYC invalid station code'
        GOTO 88000
11000  TBLLEN = NX(STNNUM)
        DO 12000 TBLLIN = 1, TBLLEN
          TBLVAL(TBLLIN,1) = YYX(TBLLIN,STNNUM)
12000  CONTINUE
        GOTO 89000
      ENDIF

C Build initial ordinate values up to highest surface level.
      TBLLIN = 0
      DO 20000 STNNUM = 1, STNMAX
        IF (BITVAL(SCC,STNNUM-1) .EQ. SET) THEN
          IF (YYX(1,STNNUM) .NE. SNDMSG) THEN
            TBLLIN = TBLLIN + 1
            TBLVAL(TBLLIN,1) = YYX(1,STNNUM)
            IF (YYX(1,STNNUM) .EQ. 0.0) GOTO 21000
          ENDIF
        ENDIF
      20000 CONTINUE

C No surface data ? zero table and return
21000 IF (TBLLIN .EQ. 0) THEN
      TBLLEN = 0
      GOTO 89000
    ENDIF

C Sort initial table lines. Descending if PR, ascending otherwise.
      IF (TBYSPC .EQ. PR) CALL BSORT(TBLVAL(1,1),TBLLIN,1)
      IF (TBYSPC .NE. PR) CALL BSORT(TBLVAL(1,1),TBLLIN,0)

C Find first ordinate value above highest surface level
      TBLLIN = TBLLIN + 1
      TBLVAL(TBLLIN,1) = 0.0
22000 IF (TBLVAL(TBLLIN,1) .GT. TBLVAL(TBLLIN-1,1)) GOTO 23000

```

```

      TBLVAL(TBLLIN,1) = TBLVAL(TBLLIN,1) + ABS(TBYINC)
      GOTO 22000

C Build table up to lowest upper level
23000 IF (TBYSPEC.EQ. PR) THEN
      TBLVAL(TBLLIN,1) = TBLVAL(TBLLIN,1) + TBYINC
      UPPER = RMN
    ELSE
      UPPER = RMX
    ENDIF

DO 24000 STNNUM = 1, STNMAX
  IF (BITVAL(SCC,STNNUM-1).EQ. SET) THEN
    IF(YXX(NX(STNNUM),STNNUM).NE. SNDMSG) THEN
      IF (TBYSPEC.EQ. PR) THEN
        UPPER = AMAX1(UPPER,YXX(NX(STNNUM),STNNUM))
      ELSE
        UPPER = AMIN1(UPPER,YXX(NX(STNNUM),STNNUM))
      ENDIF
    ENDIF
  ENDIF
24000 CONTINUE

  IF ( (UPPER.EQ. RMN).OR. (UPPER.EQ. RMX) ) THEN
    WRITE(*,90000) 'ORAWINJOB_RWPMYC upper level missing'
    GOTO 88000
  ENDIF

25000 IF ( (TBYSPEC.EQ.PR).AND. (TBLVAL(TBLLIN,1).LT.UPPER)) GOTO 26000
  IF ( (TBYSPEC.NE.PR).AND. (TBLVAL(TBLLIN,1).GT.UPPER)) GOTO 26000
  TBLLIN = TBLLIN + 1
  TBLVAL(TBLLIN,1) = TBLVAL(TBLLIN-1,1) + TBYINC
  GOTO 25000
26000 TBLLIN = TBLLIN - 1
      GOTO 89000

***ABEND***
88000 STOP 'RAWIN_RWPMYC abnormal ending'

*Terminate RWPMYC
89000 RETURN
      END

```

Input ordinate parameter

```

C*****SUBROUTINE RWPMYI (NSCC,YPM,YPMI)
* * * * *
* Input RAWIN ordinate parameter 840820 Green *
*      NSCC [I]      number of station combination codes in table *

```

```

*      YPM [I]      ordinate parameter number.      *
*      YPMI [R]     ordinate parameter increment :   *
*                  = 0   contact times               *
*                  1/2 0 height or time              *
*                  1/4 0 pressure                    *

```

```
CALLS RWOPCI
```

```

* (1) Contact point mode is only allowed if the multiple station
*     flag is clear.
* (2) No value checks are made on the ordinate increment
* (3) YPM is returned zero if NSCC is zero on entry
* * * * *

```

```
SUBROUTINE RWPMYI (NSCC,YPM,YPMI)
```

```
$INCLUDE: 'RAWINTSK.DCL'
```

```

      INTEGER      NSCC,YPM,TSTBIT
      REAL         YPMI

```

```
*Begin RWPMYI
```

```
C If no stations defined, exit.
```

```

      IF (NSCC .EQ. 0) THEN
        YPM = 0
        GOTO 89000
      ENDIF

```

```
C All enabled ordinate parameters are valid.
```

```
TSTBIT = BIT0 + BIT1
```

```
C Input the ordinate parameter and increment.
```

```

      WRITE(*,'(A)') 'Ordinate parameter specification'
      CALL RWOPCI(PMLEN,PMOPC,PMDES,PMBIT,PMTTL,TSTBIT,YPM)
      IF (YPM .NE. 0) THEN
10      WRITE(*,'(A,*)') ' Input ordinate increment : '
          READ (*,'(F13.2)') YPMI
          YPMI = ABS(YPMI)
          IF (YPMI .EQ. 0.0) THEN
            IF (MSFLG) THEN
              WRITE(*,'(A)') '0No contact point mode w/ multiple SCC'
              GOTO 10
            ENDIF
          ELSE
            IF (YPM .EQ. PR) YPMI = -YPMI
          ENDIF
      ENDIF

```

```
*Terminate RWPMYI
```

```
89000 RETURN
```

```
END
```

Input station combination code

```

C*****SUBROUTINE RWSCCI(IVSCC,ISCC)
* * * * *
* 15 July 1984 Green
*      Input station combination code
*
*Subprogram references
CALLS BITVAL in UTLTYLBR.FOR to read a bit
* * * * *

```

```

SUBROUTINE RWSCCI(IVSCC,ISCC)

```

```

$INCLUDE: 'RAWINGET.DCL'

```

```

INTEGER          IVSCC,ISCC,BITVAL
CHARACTER        BELL *1, BLANK *1, SCC(6) *1

```

```

DATA BLANK /' '/

```

```

90000 FORMAT(A,A,®)
90010 FORMAT(6A1)

```

```

*Begin RWSCCI

```

```

BELL = CHAR(7)

```

```

10000 WRITE(*,90000) ' Station code(s) ? ',BELL
READ (*,90010) (SCC(J),J=1,STNMAX)

```

```

ISCC = 0
IF (SCC(1) .EQ. BLANK) THEN
    ISCC = IVSCC
    GOTO 89000
ENDIF
ISCC = 0
DO 10900 I = 1, STNMAX
    IF (SCC(I) .EQ. BLANK) THEN
        GOTO 89000
    ENDIF
    DO 10800 J = 1, STNMAX
        IF(SCC(I).EQ.STID01(J).AND.BITVAL(IVSCC,J-1).EQ.SET) THEN
            CALL SETBIT(ISCC,J-1)
            GOTO 10900
        ENDIF
    ENDIF
10800 CONTINUE
10900 CONTINUE

```

```
*Terminate RWSCCI
89000 RETURN
END
```

Encode station ids

```
C*****SUBROUTINE RWSCCN(ISCC,SCCTXT)
* * * * *
* 15 July 1984 Green
* Encode station ids into a 12 char string
*
*Subprogram references
CALLS BITVAL in UTLTYLBR.FOR to read a bit
* * * * *
```

```
SUBROUTINE RWSCCN(ISCC,SCCTXT)
```

```
$INCLUDE:'RAWINGET.DCL'
```

```
INTEGER ISCC,NSCC,BITVAL
CHARACTER SCCTXT(12)*1, BLANK*1
DATA BLANK /' '/
```

```
90000 FORMAT(12A1)
90010 FORMAT(4A1)
```

```
*Begin RWSCCN
```

```
10000 DO 10010 I = 1, 12
      SCCTXT(I) = BLANK
10010 CONTINUE
```

```
C Count number of stations in ISCC
NSCC = 0
DO 10020 I = 1, STNMAX
  NSCC = NSCC + BITVAL(ISCC,I-1)
10020 CONTINUE
```

```
C More than 3 stations ? Use 1 char station codes
IF (NSCC .GT. 3) THEN
  K = 1
  DO 10300 I = 1, STNMAX
    IF (BITVAL(ISCC,I-1) .EQ. SET) THEN
      SCCTXT(K) = STID01(I)
      SCCTXT(K+1) = BLANK
      K = K + 2
    ENDIF
  10300 CONTINUE
```

C Only 1 station ? Use 12 char station codes

```

      ELSEIF (NSCC .EQ. 1) THEN
        DO 10040 I = 1, STNMAX
          IF (BITVAL(ISCC,I-1) .EQ. SET) THEN
            READ(STID12(I),90000) (SCCTXT(J),J=1,12)
          ENDIF
10040    CONTINUE

```

C Otherwise, use 3 char station codes

```

      ELSE
        K = 1
        DO 10050 I = 1, STNMAX
          IF (BITVAL(ISCC,I-1) .EQ. SET) THEN
            READ(STID04(I),90010) (SCCTXT(J),J=K,K+3)
            K = K + 4
          ENDIF
10050    CONTINUE
      ENDIF

```

*Terminate RWSCCN

```

89000 RETURN
      END

```

Input station combinations

```

C*****SUBROUTINE RWSTNI(MXSCC,SCC,NSCC)
* * * * *
* Input RAWIN station combinations 840820 Green *
*   MXSCC [I]      maximum number of station combination codes *
*   SCC [I]        array (MXSCC) containing station combinations *
*   NSCC [I]       number of station combination codes entered *
CALLS RWSCCI
CALLS BITVAL
CALLS SETBIT
*   All stations are considered valid, regardless of whether or *
*   not sounding data are available for the station. *
* * * * *

```

SUBROUTINE RWSTNI(MXSCC,SCC,NSCC)

\$INCLUDE: 'RAWINTSK.DCL'

```

      INTEGER MXSCC,SCC(MXSCC),NSCC
      INTEGER IVSCC,NSCCIN,BITVAL

```

```

90000 FORMAT(A)
90010 FORMAT(1X,A2,1X,A16,2X,*)
90020 FORMAT(1X,I2,*)

```

*Begin RWSTNI

```
10000 STNREQ = 0
      MSFLG = .FALSE.
      APFLG = .TRUE.
      IVSCC = BIT0 + BIT1 + BIT2 + BIT3 + BIT4 + BIT5
```

C Output instructions

```
WRITE(*,90000) '0Enter 1 char station code(s) [no delimiter]'
WRITE(*,90000) ' X in first field terminates'
WRITE(*,90000) ' ¼CR½ enables all stations'
```

C Output station codes

```
DO 10100 I = 1, STNMAX
  WRITE(*,90010) STID01(I),STID12(I)
  IF (MOD(I,2) .EQ. 0) WRITE(*,90000) ' '
10100 CONTINUE
  IF (MOD(I,2) .EQ. 0) WRITE(*,90000) ' '
```

C Input station codes

```
DO 10200 I = 1, MXSCC
  WRITE(*,90020) I
  CALL RWSCCI(IVSCC,SCC(I))
  IF (SCC(I) .EQ. 0) GOTO 10300
  NSCCIN = 0
  DO 10150 J = 1, STNMAX
    IF (BITVAL(SCC(I),J-1) .EQ. SET) THEN
      NSCCIN = NSCCIN + 1
      CALL SETBIT(STNREQ,J-1)
    ENDIF
10150 CONTINUE
```

C Set multiple station and areal parameter flags

```
IF (.NOT. MSFLG) MSFLG = NSCCIN .GE. 2
IF (APFLG) APFLG = NSCCIN .GE. 3
10200 CONTINUE
```

C Count entries

```
10300 NSCC = I - 1
```

*Terminate RWSTNI

```
89000 RETURN
```

```
END
```

Output RAWIN table

C*****SUBROUTINE RWTBLO(TSKNUM)

* * * * *

* Output RAWIN table

840820 Green *

```

*          TSKNUM [I] Current task number
CALLS LOCAL in UTLTYLBR.FOR to convert zulu to local time
CALLS GDATE in GDATE.ASM to get the current date
CALLS BITVAL in UTLTYLBR.FOR to read a bit
CALLS SETBIT in UTLTYLBR.FOR to set a bit
CALLS RWSCCN in RAWINGET.FOR to encode station code
* * * * *

```

```

SUBROUTINE RWTBLO(TSKNUM)

```

```

$INCLUDE: 'RAWINJOB.DCL'

```

```

INTEGER          TSKNUM,BITVAL,SCC

CHARACTER         SCCTXT(12)*1,TXT12(6)*12,NUL12*12
CHARACTER         TXT48(6)*48,NUL48*48,FMT*127,COLTXT(7)*12
CHARACTER         COLHDR*98,DTAFMT(7)*16

INTEGER          NPAGE,NLINE,PAGE,DAY,MON,YEAR
INTEGER          NCOL,NNUL,COL,LINE,LAT1,LAT2,LON1,LON2

```

```

DATA NUL12 /' '/
DATA NUL48 /' '/
DATA NLINE /53/

```

```

90000 FORMAT(A)
90004 FORMAT('+',A,A,*)
90005 FORMAT('+',A,A,A)
90010 FORMAT(26X,99('='))
90015 FORMAT(26X,99('-'))
90020 FORMAT(12A1)
90030 FORMAT(A4,1H:,I2,1X,I2,1X,F4.1,'N',2X,I3,1X,I2,1X,F5.1,'W',1X,
-      'ELE = ',F6.1,1X,'m',2X)
90040 FORMAT('(',I1,'(A12,1X,1H*))')
90041 FORMAT('(',9H(26X,1H*,',I1,'(A12,1X,4H1H*))')')
90050 FORMAT('(',I1,'(A12,1X,1H*),',I3,'X,1H*))')
90051 FORMAT('(',9H(26X,1H*,',I1,'(A12,7H,1X,2H*),4H',I3,'X,4H1H*))')
90060 FORMAT(26X,1H*,6(A12,2X),6X,I2,1X,1H/,1X,I2,1H*)
90070 FORMAT(26X,1H*,A30,60X,I2,1X,1H/,1X,I2,1H*)
90080 FORMAT(26X,1H*, 'Release:',1X,I8,1X,I3,'/',I3,1X,'GMT',
-      32X,'Rundate:',1X,I2,1X,A3,1X,I4,2X,'RAWIN(840820) *')
90090 FORMAT(26X,1H*,2A48,1X,1H*)
90100 FORMAT(26X,1H*,A98)
90105 FORMAT('(',A12,'*'))
90110 FORMAT(25X,*)
90120 FORMAT(1X,1H*,*)
90125 FORMAT(3X,7HMISSING,2X,*)
90130 FORMAT(13X,1H*,*)
90140 FORMAT(1X)
90150 FORMAT(26X,'Balloon drift = ',A)

```

```
91000 FORMAT(1X,'RAWINTSK_RWTBLO :',1X,A14,1X,'Unit:',1X,I2,1X,A)
```

```
*Begin RWTBLO
```

```
NPAGE = TBLLEN/NLINE + 1
IF (MOD(TBLLEN,NLINE) .EQ. 0) NPAGE = NPAGE - 1
PAGE = 0
TBLLIN = 0
CALL GDATE(YEAR,MON,DAY)
```

```
OPEN(TBLLU,FILE=TBLLFIL,STATUS='NEW')
WRITE(*,91000) TBLLFIL,TBLLU,'Opened'
WRITE(TBLLU,90004) CHAR(27),CHAR(60)
WRITE(TBLLU,90005) CHAR(27),CHAR(48),CHAR(15)
```

```
FMT = ' '
COLHDR = ' '
DO 10 I = 1, TBCMAX + 1
    DTAFMT(I) = ' '
```

```
10    CONTINUE
```

```
C Set the station combination code for the table
```

```
SCC = CLR
IF (TSNUM .EQ. MP) SCC = TBHSPC(TBHNUM)
IF (TSNUM .EQ. IP) SCC = STNREQ
```

```
C find min/max minutes
```

```
MINMN = 60
MINMX = -60
DO 20 J = 1, STNMAX
    IF (BITVAL(SCC,J-1) .EQ. SET) THEN
        IF (GETMIN(GETNUM,J) .NE. MSGSND) THEN
            MINMN = MIN0(MINMN,GETMIN(GETNUM,J))
            MINMX = MAX0(MINMX,GETMIN(GETNUM,J))
        ENDIF
    ENDIF
```

```
20    CONTINUE
```

```
C Encode station names (MP) and station locations (MP and IP)
```

```
NSCC = 0
DO 10000 STNNUM = 1, STNMAX
    TXT12(STNNUM) = NUL12
    TXT48(STNNUM) = NUL48
    IF (BITVAL(SCC,STNNUM-1) .EQ. SET) THEN
        NSCC = NSCC + 1
        IF (TSNUM .EQ. MP) THEN
            ISCC = CLR
            CALL SETBIT(ISCC,STNNUM-1)
            CALL RWSCCN(ISCC,SCCTXT)
            WRITE(TXT12(NSCC),90020) (SCCTXT(J),J=1,12)
```

```

ENDIF

LAT1 = IFIX(STLATX(1,STNNUM))
LAT2 = IFIX(STLATX(2,STNNUM))
LON1 = IFIX(STLONX(1,STNNUM))
LON2 = IFIX(STLONX(2,STNNUM))
WRITE(TXT48(NSCC),90030) STID04(STNNUM),
-                                     LAT1,LAT2,STLATX(3,STNNUM),
-                                     LON1,LON2,STLONX(3,STNNUM),
-                                     STHGTX(STNNUM)
ENDIF
10000 CONTINUE

C Encode station headers
NCOL = TBCLEN + 1
NNUL = 97 - 14 * NCOL
IF (NNUL .LE. 0) WRITE(FMT,90040) NCOL
IF (NNUL .GT. 0) WRITE(FMT,90050) NCOL,NNUL

C Column 1 = ordinate parameter
COLTXT(1) = PMCOL(TBYSPC)

DO 10200 COL = 2, NCOL

C MP column headers = parameters
IF (TSNUM .EQ. MP) THEN
    COLTXT(COL) = PMCOL(TBCSPC(COL-1))

C IP column headers = stations
ELSEIF (TSNUM .EQ. IP) THEN
    CALL RWSCCN(TBCSPC(COL-1),SCCTXT)
    WRITE(COLTXT(COL),90020) (SCCTXT(J),J=1,12)
ENDIF

10200 CONTINUE

WRITE(COLHDR,FMT) (COLTXT(J),J=1,NCOL)

C Encode data formats
WRITE(DTAFMT(1),90105) PMFMT(TBYSPC)
DO 10300 COL = 2, NCOL
    IF (TSNUM.EQ.MP) WRITE(DTAFMT(COL),90105) PMFMT(TBCSPC(COL-1))
    IF (TSNUM.EQ.IP) WRITE(DTAFMT(COL),90105) PMFMT(TBHSPC(TBHHNUM))
10300 CONTINUE

*** New page ***

C Fix top margin
20000 WRITE(TBLLU,90000) '1'

```

```

DO 20100 I = 1, 14
    WRITE(TBLLU,90000) ' '
20100 CONTINUE
    WRITE(TBLLU,90010)

C Fix counters
    PAGE = PAGE + 1
    LINE = 0

C Output title line
    IF(TSNUM.EQ.MP) WRITE(TBLLU,90060)(TXT12(J),J=1,STNMAX),PAGE,NPAGE
    IF(TSNUM.EQ.IP) WRITE(TBLLU,90070)PMDES(TBHSPC(TBHNUM)),PAGE,NPAGE

C Output release time / rundate
    WRITE(TBLLU,90080) GETDAT(GETNUM),MINMN,MINMX,
-                     DAY,MONTH(MON),YEAR

C Output station locations
    WRITE(TBLLU,90090) (TXT48(J),J=1,STNMAX)

C Output column headers
    WRITE(TBLLU,90010)
    WRITE(TBLLU,90100) COLHDR
    WRITE(TBLLU,90015)

*** New data line ***
25000 LINE = LINE + 1
    TBLLIN = TBLLIN + 1
    IF ( (LINE .LE. NLINE) .AND. (TBLLIN .LE. TBLLEN) ) THEN
        WRITE(TBLLU,90110)
        DO 25010 COL = 1, NCOL
            WRITE(TBLLU,90120)
            IF (TBLVAL(TBLLIN,COL) .EQ. SNDMSG) THEN
                WRITE(TBLLU,90125)
            ELSE
                WRITE(TBLLU,DTAFMT(COL)) TBLVAL(TBLLIN,COL)
            ENDIF
        ENDIF
25010 CONTINUE
        WRITE(TBLLU,90120)
        DO 25020 I = COL, TBCMAX + 1
            WRITE(TBLLU,90130)
25020 CONTINUE
        WRITE(TBLLU,90140)
        GOTO 25000
    ELSE

C Last line of page = task summary
    WRITE(TBLLU,90010)
    IF (APFLG .AND. BDFLG) WRITE(TBLLU,90150) 'Enabled'
    IF (APFLG .AND. (.NOT. BDFLG) ) WRITE(TBLLU,90150) 'Disabled'

```

TBLLIN = TBLLIN - 1

C Next page or quit

IF (PAGE .LT. NPAGE) GOTO 20000
ENDIF

CLOSE (TBLLU)
WRITE(*,91000) TBLFIL,TBLLU,'Closed'

*Terminate RWTBLO
89000 RETURN
END

RAWIN time conversion units

INTEGER FUNCTION RWTIM(YMDH,N)
INTEGER YMDH,N,DAYS(12),Y,M,D,H,COUNT
CHARACTER CYMDH*8
DATA DAYS /31,28,31,30,31,30,31,31,30,31,30,31/
90000 FORMAT(I8)
90010 FORMAT(4I2)

WRITE(CYMDH,90000) YMDH
READ (CYMDH,90010) Y,M,D,H
COUNT = 0
IF (MOD(Y,4) .EQ. 0) DAYS(2) = 29

DO 10 I = 1, M - 1
COUNT = COUNT + (DAYS(I) * 24 * 60)
10 CONTINUE
RWTIM = COUNT + (D - 1) * 24 * 60 + (H * 60) + N
RETURN
END

INTEGER FUNCTION RWDAT(MINUTE,N,LEAP)
INTEGER MINUTE,N,LEAP,DAYS(12),M,D,H,COUNT
DATA DAYS/31,28,31,30,31,30,31,31,30,31,30,31/

DAYS(2) = DAYS(2) + LEAP
COUNT = 0

DO 10 M = 1, 12
COUNT = COUNT + DAYS(M) * 24 * 60
IF (COUNT .GT. MINUTE) THEN
COUNT = COUNT - (DAYS(M) * 24 * 60)
GOTO 15
ENDIF
10 CONTINUE
STOP 'RAWIN_RWDAT invalid count (M)'

```

15  DO 20 D = 1, DAYS(M)
      COUNT = COUNT + 24 * 60
      IF (COUNT .GT. MINUTE) THEN
        COUNT = COUNT - (24 * 60)
        GOTO 25
      ENDIF
20  CONTINUE
      STOP 'RAWIN_RWDAT invalid count (D)'

25  DO 30 H = 1, 24
      COUNT = COUNT + 60
      IF (COUNT .GT. MINUTE) THEN
        COUNT = COUNT - 60
        GOTO 35
      ENDIF
30  CONTINUE
      STOP 'RAWIN_RWDAT invalid count (H)'

35  N = MINUTE - COUNT
      IF (N .GT. 59) THEN
        STOP 'RAWIN_RWDAT invalid count (N)'
      ENDIF
      RWDAT = H + D*100 + M*10000

89000 RETURN
      END

      INTEGER FUNCTION RWHRI(YMDH)
      INTEGER YMDH,DAYS(12),Y,M,D,H
      CHARACTER CYMDH*8
      DATA DAYS/31,28,31,30,31,30,31,31,30,31,30,31/
90000 FORMAT(I8)
90010 FORMAT(4I2)

      WRITE(CYMDH,90000) YMDH
      READ (CYMDH,90010) Y,M,D,H
      IF (MOD(Y,4) .EQ. 0) DAYS(2) = 29

      H = H + 1
      IF (H .GT. 23) THEN
        H = H - 24
        D = D + 1
        IF (D .GT. DAYS(M)) THEN
          D = 1
          M = M + 1
          IF (M .GT. 12) THEN
            M = 1
            Y = Y + 1
          ENDIF
        ENDIF
      ENDIF

```

```

      ENDIF
ENDIF
RWHRI = Y*1000000 + M*10000 + D*100 + H
RETURN
END

```

Temporary communication files

```

C*****SUBROUTINE RWTMPF(TMPNUM)
* * * * *
* RAWIN temporary file (plot communication module)      860214 ODGreen *
*
*      TMPNUM [I]      Temporary file number
* * * * *

```

```

      SUBROUTINE RWTMPF(TMPNUM)

```

```

$INCLUDE: 'RAWINJOB.DCL'

```

```

      INTEGER TMPLU,BITVAL,TMPDX,TMPNUM,SCC
      DIMENSION PMN(30),PMX(30)
      CHARACTER TMPFIL*14,SCCTXT(12)*1,C(11)*1

```

```

      DATA TMPLU/85/

```

```

$INCLUDE: 'RWPLTLIM.DCL'

```

```

90010 FORMAT(1X,'RAWIN_RWTMPF :',A14,1X,'Unit',1X,I4,1X,A)
90020 FORMAT(A2,'RAWINTMP.',3A1)

```

```

*Begin RWTMPF

```

```

C Output temporary file

```

```

10000 DO 11000 I = 1, TBCLEN

```

```

      IF (BITVAL(PFCSPC(TBHNUM),I-1) .EQ. SET) THEN

```

```

          TMPNUM = TMPNUM + 1

```

```

          CALL NCODI1(TMPNUM,C)

```

```

          WRITE(TMPFIL,90020) DRIVE(TMPDX),(C(J),J=9,11)

```

```

          OPEN(TMPLU,FILE=TMPFIL,STATUS='NEW',FORM='BINARY')

```

```

          WRITE(*,90010) TMPFIL,TMPLU,'Opened'

```

```

          WRITE(TMPLU) TBYSPEC,PMOPC(TBYSPEC),PMDES(TBYSPEC)

```

```

          IF (TSNUM .EQ. MP) THEN

```

```

              WRITE(TMPLU) TBCSPC(I),PMOPC(TBCSPC(I)),

```

```

                  PMDES(TBCSPC(I))

```

```

              CALL RWSCCN(TBHSPC(TBHNUM),SCCTXT)

```

```

              WRITE(TMPLU) TBHSPC(TBHNUM),(SCCTXT(J),J=1,12)

```

```

          ELSE

```

```

              WRITE(TMPLU) TBHSPC(TBHNUM),PMOPC(TBHSPC(TBHNUM)),

```

```

                  PMDES(TBHSPC(TBHNUM))

```

```

      CALL RWSCCN(TBCSPC(I),SCCTXT)
      WRITE(TMPLU) TBCSPC(I),(SCCTXT(J),J=1,12)
    ENDIF

    IF (TSNUM .EQ. MP) SCC = TBHSPC(TBHNUM)
    IF (TSNUM .EQ. IP) SCC = STNREQ
    MINMN = 60
    MINMX = -60
    DO 10500 J = 1, STNMAX
      IF (GETMIN(GETNUM,J) .NE. MSGSND) THEN
        IF (BITVAL(SCC,J-1) .EQ. SET) THEN
          MINMN = MIN0(MINMN,GETMIN(GETNUM,J))
          MINMX = MAX0(MINMX,GETMIN(GETNUM,J))
        ENDIF
      ENDIF
    CONTINUE
    WRITE(TMPLU) GETDAT(GETNUM),MINMN,MINMX,
      BDFLG,ASFLG,SNDSMSG
    WRITE(TMPLU) PMN,PMX

    DO 10750 K = 1, TBLLEN
      WRITE(TMPLU) TBLVAL(K,1),TBLVAL(K,I+1)
    CONTINUE
    CLOSE(TMPLU)
    WRITE(*,90010) TMPFIL,TMPLU,'Closed'
  ENDIF
11000 CONTINUE

*Terminate RWTMPF
99999 RETURN
      END

```

Set default task specifications

```

C*****SUBROUTINE RWTSKD
* * * * *
* Set RAWIN default task specs 840820 Green *
*   The default task specifies one multiple parameter table for *
*   each station. The table contains the parameters stored in the *
*   rawinsonde sounding database for contact times. *
* * * * *

```

SUBROUTINE RWTSKD

\$INCLUDE:'RAWINTSK.DCL'

*Begin RWTSKD

C Set display mode to tables and task spec to multiple parameter

DMNUM = TB
TSNUM = MP

C Table headers = 1 station per table

MSFLG = .FALSE.
TBHSPC(1) = BIT0
TBHSPC(2) = BIT1
TBHSPC(3) = BIT2
TBHSPC(4) = BIT3
TBHSPC(5) = BIT4
TBHSPC(6) = BIT5
TBHLEN = 6
STNREQ = BIT0 + BIT1 + BIT2 + BIT3 + BIT4 + BIT5

C Ordinate = time after release at contact points

TBYSPC = TM
TBYINC = 0.0

C Table columns = other sounding file parameters

APFLG = .FALSE.
TBCSPC(1) = PR
TBCSPC(2) = TE
TBCSPC(3) = TD
TBCSPC(4) = WS
TBCSPC(5) = WD
TBCLEN = 5

C Set fetch level to all data
SNDPRL = 0.0

*Terminate RWTSKD

WRITE(*,'(A)') '0Default task specifications entered'
89000 RETURN
END

Set task specifications

C*****SUBROUTINE RWTSKI

* * * * *
* Set RAWIN task specifications 840820 Green *
CALLS RWOPCI *
CALLS RWSTNI *
CALLS RWPMYI *
CALLS RWPMXI *
* * * * *

SUBROUTINE RWTSKI

\$INCLUDE: 'RAWINTSK.DCL'

```

      INTEGER  TSTBIT

90000 FORMAT(A)

*Begin RWTSKI

C Input the task specification
10000 TSTBIT = BIT0
      CALL RWOPCI(TSLEN,TSOPC,TSDES,TSBIT,TSTTL,TSTBIT,TSNUM)
      GOTO (89000,20000,30000,40000) TSNUM + 1

C TS=MP task specification = multiple parameter
C Table headers are station combination codes
C Table columns are parameter codes
20000 WRITE(*,90000) 'OMP: Table headers = station codes'
      CALL RWSTNI(TBHMAX,TBHSPC,TBHLEN)
      CALL RWPMYI(TBHLEN,TBYSPC,TBYINC)
      IF (TBYSPC .NE. 0) THEN
        WRITE(*,90000) 'OMP: Table columns = parameter codes'
      ENDIF
      CALL RWPMXI(TBYSPC,TBCMAX,TBCSPC,TBCLEN)
      IF (TBCLEN .EQ. 0) TSNUM = 0
      GOTO 89000

C TS=IP task specification = individual parameter
C Table columns are station combination codes
C Table headers are parameter codes
30000 WRITE(*,90000) 'OIP: Table columns = station codes'
      CALL RWSTNI(TBCMAX,TBCSPC,TBCLEN)
      CALL RWPMYI(TBCLEN,TBYSPC,TBYINC)
      IF (TBYSPC .NE. 0) THEN
        WRITE(*,90000) 'IP: Table headers = parameter codes'
      ENDIF
      CALL RWPMXI(TBYSPC,TBHMAX,TBHSPC,TBHLEN)
      IF (TBHLEN .EQ. 0) TSNUM = 0
      GOTO 89000

C TS=PP task specification = paired parameter
C Currently disabled
40000 CONTINUE

*Terminate RWTSKI
89000 RETURN
      END

```

Output task specifications

```
C*****SUBROUTINE RWTSKO(OUTLU,TSKNUM)
```

```

* * * * *
* Output RAWIN task specifications                                840820 Green *
*      OUTLU  [I]  output logical unit number                    *
*      TSKNUM [I]  current task number                          *
CALLS RWSCCN                                                    *
* * * * *

```

```
SUBROUTINE RWTSKO(OUTLU,TSKNUM)
```

```
$INCLUDE: 'RAWINTSK.DCL'
```

```

      INTEGER      OUTLU,TSKNUM
      CHARACTER    SCCTXT(12)*1,SCCX*12,CHR*1

```

```

90000 FORMAT(1X,'TSKNUM=',I4,1X,'STNREQ=',I4,1X,'PFCSPC=',6I4)
90010 FORMAT(1X,A2,'=',L1,*)
90020 FORMAT(1X)
90030 FORMAT(1X,I2,':',A2,'=',A2,*)
90040 FORMAT(1X,'TBHLEN=',I4,2X,'TBHSPC:',*)
90050 FORMAT(12A1)
90060 FORMAT(1X,I4,'=',A12,*)
90070 FORMAT(/,21X,*)
90080 FORMAT(1X,'TBYSPC:',1X,I4,'=',A12,2X,'TBYINC=',F8.2)
90090 FORMAT(1X,'TBCLEN=',I4,2X,'TBCSPC:',*)
90100 FORMAT(1X,I3,1X,'fetch list entries targeted on drive ',I1,'=',A1,
-          1X,'to',F7.2,1X,'mb',1X,'TTINT=',I4)
90110 FORMAT(1X,I8,3X,A12,I2,*)
90115 FORMAT(1X,6I4,1X,*)
90120 FORMAT(1X,'ENDTSK ',I4)

```

```
*Begin RWTSKO
```

```
C Output task header
```

```
10000 WRITE(OUTLU,90000) TSKNUM,STNREQ,PFCSPC
```

```
C Output processing flag status
```

```
DO 10100 I = 1, SFLEN
```

```
WRITE(OUTLU,90010) SFOPC(I),RWFLGX(I)
```

```
10100 CONTINUE
```

```
WRITE(OUTLU,90010) 'MS',MSFLG
```

```
WRITE(OUTLU,90010) 'AP',APFLG
```

```
WRITE(OUTLU,90020)
```

```
C Output display mode
```

```
IF (DMNUM.EQ. 0) THEN
```

```
WRITE(OUTLU,90030) DMNUM,'DM','__'
```

```
ELSE
```

```
WRITE(OUTLU,90030) DMNUM,'DM',DMOPC(DMNUM)
```

```
ENDIF
```

C Output task specification

```

IF (TSNUM .EQ. 0) THEN
  WRITE(OUTLU,90030) TSNUM,'TS','_'
ELSE
  WRITE(OUTLU,90030) TSNUM,'TS',TSOPC(TSNUM)
ENDIF
WRITE(OUTLU,90020)

```

C Output table headers

```

IF (TSNUM .NE. 0) THEN
  WRITE(OUTLU,90040) TBHLEN
  DO 10200 I = 1, TBHLEN
    IF (TSNUM .EQ. 1) THEN
      CALL RWSCCN(TBHSPC(I),SCCTXT)
      WRITE(SCCX,90050) (SCCTXT(J),J=1,12)
    ELSEIF (TSNUM .EQ. 2) THEN
      SCCX = PMCOL(TBHSPC(I))
    ENDIF
    WRITE(OUTLU,90060) TBHSPC(I),SCCX
    IF (MOD(I+3,3) .EQ. 0) THEN
      IF (I .EQ. TBHLEN) WRITE(OUTLU,90020)
      IF (I .NE. TBHLEN) WRITE(OUTLU,90070)
    ENDIF
  10200 CONTINUE
  IF (MOD(I+2,3) .NE. 0) WRITE(OUTLU,90020)

```

C Ordinate parameters

```

WRITE(OUTLU,90080) TBYSPEC,PMCOL(TBYSPEC),TBYINC

```

C Table columns

```

WRITE(OUTLU,90090) TBCLEN
DO 10300 I = 1, TBCLEN
  IF (TSNUM .EQ. 1) THEN
    SCCX = PMCOL(TBCSPC(I))
  ELSEIF (TSNUM .EQ. 2) THEN
    CALL RWSCCN(TBCSPC(I),SCCTXT)
    WRITE(SCCX,90050) (SCCTXT(J),J=1,12)
  ENDIF
  WRITE(OUTLU,90060) TBCSPC(I),SCCX
  IF (MOD(I+3,3) .EQ. 0) THEN
    IF (I .EQ. TBCLEN) WRITE(OUTLU,90020)
    IF (I .NE. TBCLEN) WRITE(OUTLU,90070)
  ENDIF
10300 CONTINUE
  IF (MOD(I+2,3) .NE. 0) WRITE (OUTLU,90020)
ENDIF

```

C Output sounding dates/times/stations

```

CHR = CHAR(SNDDX + 64)
WRITE(OUTLU,90100) GETLEN,SNDDX,CHR,SNDPRL,TTINT

```

```

DO 10400 I = 1, GETLEN
  CALL RWSCCN(GETSTN(I),SCCTXT)
  WRITE(SCCX,90050) (SCCTXT(J),J=1,12)
  WRITE(OUTLU,90110) GETDAT(I),SCCX,GETSTN(I)
  IF (MOD(I+3,3) .EQ. 0) WRITE(OUTLU,90020)
10400 CONTINUE
  IF (MOD(I+2,3) .NE. 0) WRITE(OUTLU,90020)
DO 10500 I = 1, GETLEN
  WRITE(OUTLU,90115) (GETMIN(I,J),J=1,6)
  IF (MOD(I+3,3) .EQ. 0) WRITE(OUTLU,90020)
10500 CONTINUE
  IF (MOD(I+2,3) .NE. 0) WRITE(OUTLU,90020)

C Output end of task banner
  WRITE(OUTLU,90120) TSKNUM

*Terminate RWTSKO
89000 RETURN
END

```

Read task specifications

```

C*****SUBROUTINE RWTSKR(TSKNUM)
* * * * *
* Read RAWIN task from taskfile 840820 Green *
CALLS NCOD11
* * * * *

```

```

SUBROUTINE RWTSKR(TSKNUM)

```

```

$INCLUDE: 'RAWINJOB.DCL'

```

```

INTEGER TSKNUM
CHARACTER FMT*80,C(11)*1

```

```

90000 FORMAT(8X,I4,8X,I4,8X,6I4)
90010 FORMAT(8(4X,L1))
90020 FORMAT(1X,I2,7X,I2)
90030 FORMAT(8X,I4)
90040 FORMAT('(21X,',I1,'(1X,I4,13X))')
90045 FORMAT(9X,I4,22X,F8.2)
90050 FORMAT(1X,I3,38X,I1,5X,F7.2,10X,I4)
90060 FORMAT('( ',I1,'(1X,I8,15X,I2))')
90065 FORMAT('( ',I1,'(1X,6I4,1X))')
91000 FORMAT('RAWINTSK.',3A1)
91010 FORMAT(A2,A12)
91020 FORMAT('ORAWIN_RWTSKR - ',1X,A14,1X,'Unit:',1X,I4,1X,A)

```

```

*Begin RWTSKR

```

C Open task file

```
CALL NCOD11(TSKNUM,C)
WRITE(TSKNAM,91000) (C(J),J=9,11)
WRITE(TSKFIL,91010) DRIVE(TSKDX),TSKNAM
OPEN(TSKLU,FILE=TSKFIL,STATUS='OLD')
WRITE(*,91020) TSKFIL,TSKLU,'Opened'
```

C Read task number and stations required

```
READ(TSKLU,90000) TSKNUM,STNREQ,PFCSPC
IF (TSKNUM.EQ. -99) GOTO 89000
```

C Read processing flags

```
READ(TSKLU,90010)(RWFLGX(J),J=1,SFLEN),MSFLG,APFLG
```

C Read display mode and task specification

```
READ(TSKLU,90020) DMNUM,TSNUM
```

C Read the table headers

```
READ(TSKLU,90030) TBHLEN
BACKSPACE TSKLU
```

```
I = 1
```

```
N = 0
```

10000 N = N + 3

```
IF (N.GT. TBHLEN) N = TBHLEN
```

```
FMT = ' '
```

```
WRITE(FMT,90040) N
```

```
READ(TSKLU,FMT) (TBHSPC(J),J=1,N)
```

```
IF (N.LT. TBHLEN) THEN
```

```
  I = I + 3
```

```
  GOTO 10000
```

```
ENDIF
```

C Read the ordinate parmeters

```
READ(TSKLU,90045) TBYSPEC,TBYINC
```

C Read the table columns

```
READ(TSKLU,90030) TBCLEN
```

```
BACKSPACE TSKLU
```

```
I = 1
```

```
N = 0
```

11000 N = N + 3

```
IF (N.GT. TBCLEN) N = TBCLEN
```

```
FMT = ' '
```

```
WRITE(FMT,90040) N
```

```
READ(TSKLU,FMT) (TBCSPC(J),J=1,N)
```

```
IF (N.LT. TBCLEN) THEN
```

```
  I = I + 3
```

```
  GOTO 11000
```

```
ENDIF
```

```

C Read fetch list, sounding drive,sounding interval
  READ(TSKLU,90050) GETLEN,SNDDX,SNDPRL,TTINT
  I = 1
  N = 0
12000 N = N + 3
      IF (N .GT. GETLEN) N = GETLEN
      FMT = ' '
      WRITE(FMT,90060) N
      READ(TSKLU,FMT) (GETDAT(J),GETSTN(J),J=I,N)
      IF (N .LT. GETLEN) THEN
        I = I + 3
        GOTO 12000
      ENDIF

C read sounding minutes
  I = 1
  N = 0
13000 N = N + 3
      IF (N .GT. GETLEN) N = GETLEN
      FMT = ' '
      WRITE(FMT,90065) N
      READ(TSKLU,FMT) ((GETMIN(J,K),K=1,STNMAX),J=I,N)
      IF (N .LT. GETLEN) THEN
        I = I + 3
        GOTO 13000
      ENDIF

C Close taskfile
  CLOSE(TSKLU)
  WRITE(*,91020) TSKFIL,TSKLU,'Closed'

*Terminate RWTSKR
89000 RETURN
END

```

Write task specifications

```

C*****SUBROUTINE RWTSKW(TSKNUM)
* * * * *
* Write RAWIN task specs to taskfile 840820 Green *
* TSKNUM [I] current task number (incremented on return) *
CALLS NCODI1 *
CALLS RWTSKO *
* Taskfile opened/closed locally on unit TSKLU *
* * * * *

SUBROUTINE RWTSKW(TSKNUM)

```

```
$INCLUDE: 'RAWINTSK.DCL'
```

```
      INTEGER      TSKNUM
      CHARACTER    C(11)*1
```

```
90000 FORMAT(A)
90010 FORMAT('RAWINTSK.',3A1)
90020 FORMAT(A2,A12)
90030 FORMAT('ORAWIN_RWTSKW - ',1X,A14,1X,'Unit',1X,I4,1X,A)
```

```
*Begin RWTSKW
```

```
C Only output to disk if task description complete
```

```
  IF (DMNUM .EQ. 0) THEN
    WRITE(*,90000) '0Display mode (DM) not set -- no write'
    GOTO 89000
  ENDIF
```

```
  IF (TSNUM .EQ. 0) THEN
    WRITE(*,90000) '0Task specification (TS) not set -- no write'
    GOTO 89000
  ENDIF
```

```
C Task description complete -- write out to jobfile
```

```
  CALL NCOD11(TSKNUM,C)
  WRITE(TSKNAM,90010) (C(J),J=9,11)
  WRITE(TSKFIL,90020) DRIVE(TSKDX),TSKNAM
  OPEN(TSKLU,FILE=TSKFIL,STATUS='NEW')
  WRITE(*,90030) TSKFIL,TSKLU,'Opened'
  CALL RWTSKO(TSKLU,TSKNUM)
  CLOSE(TSKLU)
  WRITE(*,90030) TSKFIL,TSKLU,'Closed'
  TSKNUM = TSKNUM + 1
```

```
*Terminate RWTSKW
```

```
89000 RETURN
      END
```

RAWIN task building initialization

```
C*****SUBROUTINE RWTSKZ
```

```
*****
* RAWINT initialization                                     840820 Green *
CALLS RWGETZ                                              *
CALLS DFDRV                                              *
*****
```

```
      SUBROUTINE RWTSKZ
```

```
$INCLUDE: 'RAWINTSK.DCL'
```

```
CHARACTER      BELL*1, CHR*1
```

```
*Begin RWTSKZ
```

```
BELL = CHAR(7)
```

```
C-----RAWINGET common initialization-----
```

```
CALL RWGETZ
```

```
C-----Processing option codes-----
```

```
POLEN      = 12
```

```
DO 10 I = 1, POLEN
```

```
POOPCN(I) = I
```

```
10 CONTINUE
```

```
POOPC(1) = 'NP'
```

```
POOPC(2) = 'SF'
```

```
POOPC(3) = 'SP'
```

```
POOPC(4) = 'SX'
```

```
POOPC(5) = 'DM'
```

```
POOPC(6) = 'TS'
```

```
POOPC(7) = 'PF'
```

```
POOPC(8) = 'DF'
```

```
POOPC(9) = 'DT'
```

```
POOPC(10) = 'WT'
```

```
POOPC(11) = 'TI'
```

```
POOPC(12) = 'EN'
```

```
C      set bit 0 to disable PO code / other bits not used
```

```
POBIT(1) = CLR
```

```
POBIT(2) = CLR
```

```
POBIT(3) = CLR
```

```
POBIT(4) = CLR
```

```
POBIT(5) = CLR
```

```
POBIT(6) = CLR
```

```
POBIT(7) = CLR
```

```
POBIT(8) = CLR
```

```
POBIT(9) = CLR
```

```
POBIT(10) = CLR
```

```
POBIT(11) = CLR
```

```
POBIT(12) = CLR
```

```
PODES(1) = 'new processing period'
```

```
PODES(2) = 'set processing flags'
```

```
PODES(3) = 'set fetch upper pressure level'
```

```
PODES(4) = 'set sounding file drive'
```

```
PODES(5) = 'set display mode'
```

```

PODES(6) = 'set task specification      '
PODES(7) = 'set plot format            '
PODES(8) = 'set default task specification'
PODES(9) = 'display task specification  '
PODES(10) = 'write task to jobfile      '
PODES(11) = 'set sounding time interval '
PODES(12) = 'end task description       '
POTTTL    = 'Processing option code    '

```

C-----processing flag codes-----

```
SFLEN      = 6
```

```
DO 20 I = 1, SFLEN
  SFOPCN(I) = I
```

20 CONTINUE

```

SFOPC(1) = 'BD'
SFOPC(2) = 'PS'
SFOPC(3) = 'SV'
SFOPC(4) = 'IO'
SFOPC(5) = 'FD'
SFOPC(6) = 'AS'

```

C set bit 0 to disable flag / other bits not used

```

SFBIT(1) = BIT0
SFBIT(2) = CLR
SFBIT(3) = BIT0
SFBIT(4) = BIT0
SFBIT(5) = CLR
SFBIT(6) = CLR

```

```

BDFLG = .FALSE.
PSFLG = .FALSE.
SVFLG = .FALSE.
IOFLG = .TRUE.
FDFLG = .FALSE.
ASFLG = .FALSE.

```

```

SFDES(1) = 'balloon drift calculations  '
SFDES(2) = 'pause for diskette exchange '
SFDES(3) = 'save output files           '
SFDES(4) = 'immediate output mode       '
SFDES(5) = 'file data dump              '
SFDES(6) = 'ordinate autoscale flag     '
SFTTTL    = 'Processing flag code       '

```

C-----display mode codes-----

```
DMLEN      = 3
```

```
DO 30 I = 1, DMLEN
```

```

      DMOPCN(I) = I
30    CONTINUE

```

```

      DMOPC(1)  = 'TB'
      DMOPC(2)  = 'PL'
      DMOPC(3)  = 'BO'

```

```

C          set bit 0 to disable DM code / other bits not used

```

```

      DMBIT(1)  = CLR
      DMBIT(2)  = CLR
      DMBIT(3)  = CLR

```

```

      DMDDES(1) = 'tables only'      '
      DMDDES(2) = 'plots only'      '
      DMDDES(3) = 'both tables and plots' '
      DMTTL     = 'Display mode code' '

```

```

C-----task specification codes-----

```

```

      TSLEN     = 3

```

```

      DO 40 I = 1, TSLEN
        TSOPCN(I) = I

```

```

40    CONTINUE

```

```

      TSOPC(1)  = 'MP'
      TSOPC(2)  = 'IP'
      TSOPC(3)  = 'PP'

```

```

C          set bit 0 to disable TS code / other bits not used

```

```

      TSBIT(1)  = CLR
      TSBIT(2)  = CLR
      TSBIT(3)  = BIT0

```

```

      TSDES(1)  = 'multiple parameter' '
      TSDES(2)  = 'individual parameter' '
      TSDES(3)  = 'paired parameter'    '
      TSTTL     = 'Task specification code' '

```

```

C-----plot format codes-----

```

```

      PFLEN     = 2

```

```

      DO 50 I = 1, PFLEN
        PFOPCN(I) = I

```

```

50    CONTINUE

```

```

      PFOPC(1)  = 'IX'
      PFOPC(2)  = 'MX'

```

```

C          set bit 0 to disable PF code / other bits not used

```

```

      PFBIT(1)  = BIT0

```

```
PFBIT(2) = BIT0
```

```
PFDES(1) = 'individual abscissa      '
```

```
PFDES(2) = 'multiple abscissa      '
```

```
PFTTL   = 'Plot format code      '
```

```
C-----parameter codes-----
```

```
PMLN    = 30
```

```
DO 60 I = 1, PMLN
```

```
    PMOPCN(I) = I
```

```
60    CONTINUE
```

```
    PMOPC(1) = 'PR'
```

```
    PMOPC(2) = 'HM'
```

```
    PMOPC(3) = 'ZM'
```

```
    PMOPC(4) = 'HF'
```

```
    PMOPC(5) = 'ZF'
```

```
    PMOPC(6) = 'TM'
```

```
    PMOPC(7) = 'TE'
```

```
    PMOPC(8) = 'TD'
```

```
    PMOPC(9) = 'TV'
```

```
    PMOPC(10) = 'TP'
```

```
    PMOPC(11) = 'TQ'
```

```
    PMOPC(12) = 'WS'
```

```
    PMOPC(13) = 'WD'
```

```
    PMOPC(14) = 'UU'
```

```
    PMOPC(15) = 'VV'
```

```
    PMOPC(16) = 'EA'
```

```
    PMOPC(17) = 'ES'
```

```
    PMOPC(18) = 'MR'
```

```
    PMOPC(19) = 'SH'
```

```
    PMOPC(20) = 'RH'
```

```
    PMOPC(21) = 'CV'
```

```
    PMOPC(22) = 'RV'
```

```
    PMOPC(23) = 'AV'
```

```
    PMOPC(24) = 'MC'
```

```
    PMOPC(25) = 'AE'
```

```
    PMOPC(26) = 'PE'
```

```
    PMOPC(27) = 'KE'
```

```
    PMOPC(28) = 'XD'
```

```
    PMOPC(29) = 'YD'
```

```
    PMOPC(30) = 'WW'
```

```
C      set bit 0 to disable PM code
C      set bit 1 to disable ordinate parameter
C      set bit 2 to disable point parameter
C      other bits not used
```

```
PMBIT(1) = CLR
```

```
PMBIT(2) = CLR
```

```
PMBIT(3) = CLR
```

```

PMBIT(4)   = CLR
PMBIT(5)   = CLR
PMBIT(6)   = CLR
PMBIT(7)   = BIT1
PMBIT(8)   = BIT1
PMBIT(9)   = BIT1
PMBIT(10)  = BIT1
PMBIT(11)  = BIT1
PMBIT(12)  = BIT1
PMBIT(13)  = BIT1
PMBIT(14)  = BIT1
PMBIT(15)  = BIT1
PMBIT(16)  = BIT1
PMBIT(17)  = BIT1
PMBIT(18)  = BIT1
PMBIT(19)  = BIT1
PMBIT(20)  = BIT1
PMBIT(21)  = BIT1 + BIT2
PMBIT(22)  = BIT1 + BIT2
PMBIT(23)  = BIT0 + BIT1 + BIT2
PMBIT(24)  = BIT1 + BIT2
PMBIT(25)  = BIT1
PMBIT(26)  = BIT0 + BIT1
PMBIT(27)  = BIT0 + BIT1
PMBIT(28)  = BIT0 + BIT1
PMBIT(29)  = BIT0 + BIT1
PMBIT(30)  = BIT1 + BIT2

PMDES(1)   = '      PRESSURE [MB]      '
PMDES(2)   = '      HEIGHT - MSL [M]    '
PMDES(3)   = '      HEIGHT - AGL [M]    '
PMDES(4)   = '      HEIGHT - MSL [FT]   '
PMDES(5)   = '      HEIGHT - AGL [FT]   '
PMDES(6)   = '      TIME AFTER RELEASE [MIN] '
PMDES(7)   = '      TEMPERATURE [C]     '
PMDES(8)   = '      DEWPOINT [C]        '
PMDES(9)   = '      VIRTUAL TEMPERATURE [K] '
PMDES(10)  = '      POTENTIAL TEMPERATURE [K] '
PMDES(11)  = '      EQUIVALENT POT. TEMP. [K] '
PMDES(12)  = '      WIND SPEED [M/S]     '
PMDES(13)  = '      WIND DIRECTION [DEG]  '
PMDES(14)  = '      EASTWARD WIND COMPONENT [M/S] '
PMDES(15)  = '      NORTHWARD WIND COMPONENT [M/S] '
PMDES(16)  = '      AMBIENT VAPOR PRESSURE [MB] '
PMDES(17)  = '      SATURATION VAPOR PRESSURE [MB] '
PMDES(18)  = '      MIXING RATIO [G/G]    '
PMDES(19)  = '      SPECIFIC HUMIDITY [G/G] '
PMDES(20)  = '      RELATIVE HUMIDITY [G/G] '
PMDES(21)  = '      KINEMATIC CONVERGENCE [1/S] '
PMDES(22)  = '      RELATIVE VORTICITY [1/S] '

```

```

PMDES(23) = ' ABSOLUTE VORTICITY [1/S] '
PMDES(24) = ' MOISTURE CONVERGENCE [G/M3-S] '
PMDES(25) = ' ATMOSPHERIC ENERGY [CAL/G] '
PMDES(26) = ' POTENTIAL ENERGY '
PMDES(27) = ' KINETIC ENERGY '
PMDES(28) = ' EASTWARD DRIFT COMP. [M] '
PMDES(29) = ' NORTHWARD DRIFT COMP. [M] '
PMDES(30) = ' VERTICAL VELOCITY [M/S] '
PMTTL     = ' Parameter code '

```

```

PMCOL(1)  = ' PRES [mb] '
PMCOL(2)  = ' MSL HGT [m] '
PMCOL(3)  = ' AGL HGT [m] '
PMCOL(4)  = ' MSL HGT [ft] '
PMCOL(5)  = ' AGL HGT [ft] '
PMCOL(6)  = ' TIME [min] '
PMCOL(7)  = ' TEMP [C] '
PMCOL(8)  = ' DEWPT [C] '
PMCOL(9)  = ' VIRT TE [K] '
PMCOL(10) = ' POT TE [K] '
PMCOL(11) = ' EQPOT TE [K] '
PMCOL(12) = ' WSPD [m/s] '
PMCOL(13) = ' WDIR [deg] '
PMCOL(14) = ' U [m/s] '
PMCOL(15) = ' V [m/s] '
PMCOL(16) = ' VAP PR [mb] '
PMCOL(17) = ' SAT VP [mb] '
PMCOL(18) = ' MIXR [g/g] '
PMCOL(19) = ' SP HUM [g/g] '
PMCOL(20) = ' REL HUM [%] '
PMCOL(21) = ' CONV [1/s] '
PMCOL(22) = ' R VORT [1/s] '
PMCOL(23) = ' A VORT [1/s] '
PMCOL(24) = ' MCON[g/m3-s] '
PMCOL(25) = ' ATM E[cal/g] '
PMCOL(26) = ' POT E '
PMCOL(27) = ' KIN E '
PMCOL(28) = ' E DRIVE [m] '
PMCOL(29) = ' N DRIVE [m] '
PMCOL(30) = ' W [m/s] '

```

```

PMFMT(1) = '3X,F6.1,3X,'
PMFMT(2) = '2X,F8.1,2X,'
PMFMT(3) = '2X,F8.1,2X,'
PMFMT(4) = '2X,F8.1,2X,'
PMFMT(5) = '2X,F8.1,2X,'
PMFMT(6) = '3X,F6.1,3X,'
PMFMT(7) = '3X,F6.1,3X,'
PMFMT(8) = '3X,F6.1,3X,'
PMFMT(9) = '3X,F6.1,3X,'

```

```

PMFMT(10) = '3X,F6.1,3X,'
PMFMT(11) = '3X,F6.1,3X,'
PMFMT(12) = '3X,F6.1,3X,'
PMFMT(13) = '3X,F6.1,3X,'
PMFMT(14) = '3X,F6.2,3X,'
PMFMT(15) = '3X,F6.2,3X,'
PMFMT(16) = '3X,F6.1,3X,'
PMFMT(17) = '3X,F6.1,3X,'
PMFMT(18) = '3X,F6.4,3X,'
PMFMT(19) = '3X,F6.4,3X,'
PMFMT(20) = '3X,F6.1,3X,'
PMFMT(21) = '1X,E10.4,1X,'
PMFMT(22) = '1X,E10.4,1X,'
PMFMT(23) = '1X,E10.4,1X,'
PMFMT(24) = '1X,E10.4,1X,'
PMFMT(25) = '1X,E10.4,1X,'
PMFMT(26) = '1X,E10.4,1X,'
PMFMT(27) = '1X,E10.4,1X,'
PMFMT(28) = '2X,F8.1,2X,'
PMFMT(29) = '2X,F8.1,2X,'
PMFMT(30) = '1X,E10.4,1X,'

```

C-----task specifications-----

```

TBHMAX = 6
TBCMAX = 6
TSKLU  = 82
CALL DFDRV(I)
TSKDX  = I
TTINT  = 55

```

```

*Terminate RWTASKZ
89000 RETURN
END

```

Build master ordinate array

C*****SUBROUTINE RWYYXB

```

* * * * *
* Build RAWIN master ordinate array 840820 Green *
* * * * *

```

SUBROUTINE RWYYXB

\$INCLUDE: 'RAWINJOB.DCL'

```

REAL      FT
DATA FT /3.28084/

```

*Begin RWYYXB

```

DO 10 J = 1, STNMAX
DO 10 I = 1, SNDMAX
    YYX(I,J) = SNDMSG
10    CONTINUE

C Transfer data from master array to primary ordinate array
10000 DO 19000 STNNUM = 1, STNMAX
    DO 18000 I = 1, NX(STNNUM)
        IF (TBYSPC .EQ. PR) THEN
            YYX(I,STNNUM) = PRX(I,STNNUM)
        ELSEIF (TBYSPC .EQ. HM) THEN
            YYX(I,STNNUM) = HMX(I,STNNUM)
        ELSEIF (TBYSPC .EQ. ZM) THEN
            YYX(I,STNNUM) = HMX(I,STNNUM) - STHGTX(STNNUM)
        ELSEIF (TBYSPC .EQ. HF) THEN
            YYX(I,STNNUM) = HMX(I,STNNUM) * FT
        ELSEIF (TBYSPC .EQ. ZF) THEN
            YYX(I,STNNUM) = (HMX(I,STNNUM) - STHGTX(STNNUM)) * FT
        ELSEIF (TBYSPC .EQ. TM) THEN
            YYX(I,STNNUM) = TMX(I,STNNUM)
        ENDIF
    18000 CONTINUE
19000 CONTINUE

*Terminate RWYYXB
89000 RETURN
END

```

Directory search

```

C-----
C*****SUBROUTINE SEARCH (FILNAM,DRIVE,FOUND)
C-----
C      SEARCH PERFORMS A DIRECTORY SEARCH ON THE SPECIFIED DRIVE
C      FOR THE INDICATED FILE
C ENTRY
C      FILNAM [C*12]  FILE NAME (NO DRIVE SPEC)  EG: FILENAME.EXT
C                     (LEFT-JUSTIFIED WITH TRAILING BLANKS)
C      DRIVE  [I]     DISK DRIVE NUMBER (1=A,2=B,ETC. 0=DEFAULT)
C                     ***IT IS THE USER'S RESPONSIBILITY TO ENSURE THAT
C                     THE DISK DRIVE NUMBER IS VALID.
C EXIT
C      FOUND  [I]     = 0   FILE FOUND
C                     = 255 FILE NOT FOUND
C SUBPROGRAM REFERENCES
CALLS LOOK
C-----
SUBROUTINE SEARCH (FILNAM,DRIVE,FOUND)

```

```

CHARACTER      FILNAM*12,FN(8)*1,FX(3)*1,F(12)*1,FCB*37
INTEGER        DRIVE,FOUND

C
91000 FORMAT(12A1)
91040 FORMAT(' ',8A1,3A1,25(' '))
C
      FOUND = 0
C      DECODE FILE NAME. PAD FIELDS W/BLANKS WHERE NECESSARY
1000 READ(FILNAM,91000) F
      DO 1010 I = 1, 8
        IF (F(I) .EQ. ' ') THEN
          GOTO 1020
        ELSE
          FN(I) = F(I)
        ENDIF
1010 CONTINUE
1020 DO 1030 J = I, 8
      FN(J) = ' '
1030 CONTINUE
C      DECODE FILE EXTENT
      I = I + 1
      DO 1040 J = I, I + 2
        FX(J-I+1) = F(J)
1040 CONTINUE
C      BUILD FCB, DO SEARCH, THEN QUIT
      FCB = '
      WRITE(FCB,91040) FN,FX
      CALL LOOK(FCB,DRIVE,FOUND)
      RETURN
      END

```

Convert wind components into direction and speed

```

C*****SUBROUTINE WIND(UU,VV,MSNG,DD,FF)
C-----HEIMBACH, SEPTEMBER, 1983-----
      SUBROUTINE WIND(UU,VV,MSNG,DD,FF)
C...SUBROUTINE TO CONVERT WIND COMPONENTS INTO DIRECTION AND SPEED.
C...ARGUMENTS: UU = EASTWARD COMPONENT,
C               VV = NORTHWARD COMPONENT,
C               MSNG = MISSING CODE,
C               DD = RETURNED DIRECTION, AND
C               FF = RETURNED SPEED

      REAL MSNG
      IF(UU.EQ.MSNG.OR.VV.EQ.MSNG) THEN
        DD=MSNG
        FF=MSNG
        RETURN
      ENDIF
      DD=ATAN2(VV,UU)

```

```

DD=270.-DD*57.29578
IF(DD.GT.360.) DD=DD-360.
FF=SQRT((UU*UU)+(VV*VV))
RETURN
END

```

Real linear interpolation

```

C*****REAL FUNCTION XINTRP(Y1,Y2,Y,X1,X2,M)
* * * * *
*
*
* * * * *
REAL FUNCTION XINTRP(Y1,Y2,Y,X1,X2,M)
REAL Y1,Y2,Y,X1,X2,M

IF (Y1.EQ.M.OR.Y2.EQ.M.OR.Y.EQ.M.OR.X1.EQ.M.OR.X2.EQ.M) THEN
  XINTRP = M
ELSE
  XINTRP = X2 - (Y2 - Y) / (Y2 - Y1) * (X2 - X1)
ENDIF

RETURN
END

```

Miscellaneous FORTRAN routines

```

C-----
C*****INTEGER FUNCTION ZULU(DATLOC,HROFF)
C-----
C 18 APRIL 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
C      ZULU RETURNS THE ZULU DATE/TIME CORRESPONDING TO THE LOCAL TIME
C ENTRY
C      DATLOC [I]  LOCAL DATE/TIME IN YYMMDDHH FORMAT
C      HROFF  [I]  HOUR OFFSET TO ZULU
C RETURN
C      ZULU  [I]  ZULU DATE/TIME IN YYMMDDHH FORMAT
*****NO VALUE CHECKS ON ENTRY DATE

```

```

C-----
      INTEGER FUNCTION ZULU(DATLOC,HROFF)
      INTEGER          DATLOC,HROFF,YY,MM,DD,HH,DAYS(12)
      CHARACTER        LOCDAT *8

C
      DATA  DAYS  /31,28,31,30,31,30,31,31,30,31,30,31/

C
      EXTRACT YEAR,MONTH,DAY,HOUR, FROM DATLOC
      WRITE(LOCDAT,'(I8)') DATLOC
      READ(LOCDAT,'(4I2)') YY,MM,DD,HH

C
      ADJUST FEB FOR LEAP YEAR (GOOD TO YEAR 2000)
      IF (MOD(YY,4) .EQ. 0) DAYS(2) = 29

C
      ADD OFFSET TO LOCAL TIME
      HH = HH + HROFF

C
      ADJUST HOUR/DAY
      IF (HH .GE. 24) THEN
        HH = HH - 24
        DD = DD + 1
      ENDIF

C
      ADJUST DAY/MONTH
      IF (DD .GT. DAYS(MM)) THEN
        DD = 1
        MM = MM + 1
      ENDIF

C
      ADJUST MONTH/YEAR
      IF (MM .GT. 12) THEN
        MM = 1
        YY = YY + 1
      ENDIF

C
      DONE. RETURN ZULU DATE
      ZULU = YY * 1000000 + MM * 10000 + DD * 100 + HH

C
      RETURN
      END

```

```

C-----
C*****INTEGER FUNCTION LOCAL (DATZUL,HROFF)
C-----

```

```

C 01 JUNE 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
C      LOCAL RETURNS THE LOCAL TIME CORRESPONDING TO THE ZULU TIME
C ENTRY
C      DATZUL  [I]  ZULU DATE/TIME IN YYMMDDHH FORMAT
C      HROFF   [I]  LOCAL HOUR OFFSET TO ZULU
C EXIT
C      LOCAL   [I]  LOCAL DATE/TIME IN YYMMDDHH FORMAT
C      ENTRY ARGUMENTS UNCHANGED
*****NO VALUE CHECKS ON ENTRY DATE
C-----
      INTEGER FUNCTION LOCAL (DATZUL,HROFF)
      INTEGER      DATZUL,HROFF,YY,MM,DD,HH,DAYS(12)
      CHARACTER    ZULDAT *8
C
      DATA DAYS /31,28,31,30,31,30,31,31,30,31,30,31/
C
*BEGIN LOCAL
C      EXTRACT YEAR,MONTH,DAY,HOURL FROM DATZUL
      WRITE(ZULDAT,'(I8)') DATZUL
      READ (ZULDAT,'(4I2)') YY,MM,DD,HH
C      ADJUST FEB FOR LEAP YEAR
      IF (MOD(YY,4) .EQ. 0) DAYS(2) = 29
C      SUBTRACT OFFSET FROM LOCAL TIME
      HH = HH - HROFF
C      ADJUST HOUR/DAY
      IF (HH .LT. 0) THEN
          HH = HH + 24
          DD = DD - 1
      ENDIF
C      ADJUST DAY/MONTH/YEAR
      IF (DD .LE. 0) THEN
          MM = MM - 1
          IF (MM .LE. 0) THEN
              YY = YY - 1
              MM = 12 - MM
          ENDIF
          DD = DAYS(MM)
      ENDIF
C
*TERMINATE LOCAL
      LOCAL = YY * 1000000 + MM * 10000 + DD * 100 + HH
C      RETURN
      END
C-----
C*****SUBROUTINE SETBIT (I,J)
C-----
C 01 JUNE 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
C      SETBIT SETS THE INDICATED BIT
C ENTRY
C      I [I]  TARGET VALUE

```

```

C      J [I]  BIT NUMBER TO SET
C EXIT
C      VALUE I WITH BIT J SET
C SUBPROGRAM REFERENCES
CALLS BITVAL

```

```

C-----
      SUBROUTINE SETBIT (I,J)
      INTEGER          I,J,BITVAL
C
      IF (BITVAL(I,J) .EQ. 0)  I = I + 2 ** J
C      RETURN
      END

```

```

C-----
C*****INTEGER FUNCTION BITVAL (I,J)
C-----

```

```

C 01 JUNE 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
C      BITVAL RETURNS THE BIT SETTING FOR THE INDICATED BIT
C ENTRY
C      I [I]  VALUE TO TEST
C      J [I]  BIT NUMBER TO TEST
C EXIT
C      BITVAL [I]  =0 BIT NOT SET
C                  =1 BIT SET
C      ENTRY ARGUMENTS UNCHANGED

```

```

C-----
      INTEGER FUNCTION BITVAL (I,J)
      INTEGER          I, J

```

```

C
      IV = I
      IB = J
10      IF (IB .EQ. 0)  GOTO 20
          IV = IV/2
          IB = IB-1
          GOTO 10
20      BITVAL = MOD(IV+2,2)
C      RETURN
      END

```

```

C*****SUBROUTINE TIMINC(YMD,HNS,TI,TU)

```

```

* * * * *
* Time increment / decrement (to Feb 28, 2000)      840810 Green *
*
* Entry:
*      YMD [I]  year,month,day in YYMMDD format
*      HNS [I]  hour,minute,second in HHNNSS format
*      TI [I]  time increment ( ¼0 means decrement )
*      TU [I]  time increment units :
*                =1 sec =2 min =3 hr =4 day
*      **no value tests on entry arguments
*
* Exit:

```

```

*      YMD,HNS incremented date/time in entry formats      *
*      TI,TU returned as entry values                      *
* * * * *

```

```

SUBROUTINE TIMINC(YMD,HNS,TI,TU)

```

```

      INTEGER      YMD,HNS,TI,TU
      INTEGER      Y,M,D,H,N,S,TIM(5),TMX(3),DAYS(12)
      CHARACTER    DCODE*6

```

```

      EQUIVALENCE  (TIM(1),S), (TIM(2),N), (TIM(3),H)
      EQUIVALENCE  (TIM(4),D), (TIM(5),M)

```

```

      DATA TMX /60,60,24/

```

```

      DATA DAYS /31,28,31,30,31,30,31,31,30,31,30,31/

```

```

90000 FORMAT(I6)
90010 FORMAT(3I2)

```

```

*Begin TIMINC

```

```

C Breakdown date/time

```

```

      WRITE(DCODE,90000) YMD
      READ (DCODE,90010) Y,M,D
      WRITE(DCODE,90000) HNS
      READ (DCODE,90010) H,N,S

```

```

C Perform the increment/decrement

```

```

      TIM(TU) = TIM(TU) + TI

```

```

C Adjust seconds, minutes, and hours

```

```

      DO 20 I = 1, 3
10      IF (TIM(I) .GE. TMX(I)) THEN
          TIM(I) = TIM(I) - TMX(I)
          TIM(I+1) = TIM(I+1) + 1
          GOTO 10
        ELSEIF (TIM(I) .LT. 0) THEN
          TIM(I) = TIM(I) + TMX(I)
          TIM(I+1) = TIM(I+1) - 1
          GOTO 10
      ENDIF
20      CONTINUE

```

```

C Adjust day, month, and year

```

```

30      IF (MOD(Y,4) .EQ. 0) DAYS(2) = 29
      IF (MOD(Y,4) .NE. 0) DAYS(2) = 28
      IF (D .GT. DAYS(M)) THEN
          D = D - DAYS(M)
          M = M + 1
          IF (M .GT. 12) THEN

```

```

        M = M - 12
        Y = Y + 1
    ENDIF
    GOTO 30
ELSEIF (D .LE. 0) THEN
    M = M - 1
    IF (M .LE. 0) THEN
        M = M + 12
        Y = Y - 1
    ENDIF
    D = D + DAYS(M)
    GOTO 30
ENDIF

```

C Format the date/time

```

    YMD = Y*10000 + M*100 + D
    HNS = H*10000 + N*100 + S

```

* Terminate TIMINC

RETURN

END

C*****INTEGER FUNCTION MINYR(YMD,HN)

```

* * * * *
* Calculate minutes after year's start 840810 Green *
*
* Entry:
*   YMD [I] year,month,day in YYMMDD format
*   HN [I] hour,minute in HHNN format
*   **no value tests on entry arguments
*
* Exit:
*   MINYR [I] time in minutes after year's start
*   YMD,HN returned as entry values
* * * * *

```

INTEGER FUNCTION MINYR(YMD,HN)

```

INTEGER      YMD,HN
INTEGER      Y,M,D,H,N,DAYS(12)
CHARACTER    DCODE*6

```

DATA DAYS /31,28,31,30,31,30,31,31,30,31,30,31/

90000 FORMAT(I6)

90010 FORMAT(3I2)

*Begin MINYR

C Breakdown date/time

WRITE(DCODE,90000) YMD

```

READ (DCODE,90010) Y,M,D
H = HN / 100
N = MOD(HN,100)

```

```

C Leap year adjustment
  IF (MOD(Y,4) .EQ. 0) DAYS(2) = 29
  IF (MOD(Y,4) .NE. 0) DAYS(2) = 28

```

```

C Add up the minutes
  DO 10 I = 1, M - 1
    D = D + DAYS(I)
10  CONTINUE
  MINYR = (D-1)*1440 + H*60 + N

```

```

*Terminate MINYR
  RETURN
  END

```

```

C*****SUBROUTINE YRMIN(NN,MD,HN,LEAP)
* * * * *
* Find date from minutes after year's start      840810 Green *
*
* Entry:
*   NN   [I] minutes after year's start
*   LEAP [I] =0 not leap year 1/20 leap year
*   **no value tests on entry arguments
*
* Exit:
*   MD   [I] month,day in MMDD format
*   HN   [I] hour,minute in HHNN format
*   NN,LEAP returned as entry values
* * * * *

```

```

SUBROUTINE YRMIN(NN,MD,HN,LEAP)

```

```

  INTEGER      NN,MD,HN,LEAP
  INTEGER      M,D,H,N,DAYS(12),COUNT

```

```

  DATA DAYS /31,28,31,30,31,30,31,31,30,31,30,31/

```

```

*Begin YRMIN

```

```

C Leap year adjustment
  IF (LEAP .EQ. 0) DAYS(2) = 28
  IF (LEAP .NE. 0) DAYS(2) = 29

```

```

C Get month
  COUNT = 0
  DO 10 M = 1, 11
    COUNT = COUNT + (DAYS(M) * 1440)

```

```
      IF (COUNT .GT. NN) THEN
        COUNT = COUNT - DAYS(M) * 1440
        GOTO 15
      ENDIF
10    CONTINUE
15    COUNT = NN - COUNT

C Get day, hour, and minute
      D = COUNT / 1440 + 1
      COUNT = MOD(COUNT, 1440)
      H = COUNT / 60
      N = MOD(COUNT, 60)

C Format date/time
      MD = M*100 + D
      HN = H*100 + N

*Terminate YRMIN
      RETURN
      END
```

Miscellaneous MACRO ASSEMBLER routines

;SOURCE FILE: ERASE.ASM

; 01 JUNE 1983 ORVILLE D. GREEN IBM/PC DOS 2.0
; ERASE CLEARS THE CONSOLE SCREEN

; CALLING SEQUENCE: CALL ERASE

;SET UP COMMUNICATION FRAME WITH CALLER

FRAME	STRUC		
ERASE_BP	DW	?	;CALLER'S BP
ERASE_RET	DD	?	;RETURN ADDRESS
FRAME	ENDS		

;REQUIRED FOR FORTRAN CALL

CSEG	SEGMENT	'CODE'	
DGROUP	GROUP	DATA	
	ASSUME	CS:CSEG,DS:DGROUP,ES:DGROUP,SS:DGROUP	
ERASE	PROC	FAR	
	PUBLIC	ERASE	
	PUSH	BP	;SAVE CALLER'S BP
	MOV	BP,SP	;SET FRAME BASE

;PROCESS ERASE REQUEST

	MOV	CX,0	;SPECIFY POS(0,0)
	MOV	DX,2479H	;SPECIFY POS(24,79)
	MOV	BH,7	;NORMAL ATTRIBUTE = 7
	MOV	AX,600H	;SCROLL ENTIRE WINDOW
	INT	10H	;CALL BIOS TO CLEAR SCREEN
	POP	BP	;RESTORE CALLER'S BP
	RET		;FREE 2 PARMS ON RETURN
ERASE	ENDP		
CSEG	ENDS		
	END		

; SOURCE FILE: GDATE.ASM

; 01 JUNE 1983 ORVILLE D. GREEN IBM/PC DOS 2.0
; GDATE RETURNS THE CURRENT DATE

; CALLING SEQUENCE : CALL GDATE (IYR,IMON,IDAY)

	IYR	[INTEGER *2]	: YEAR (1980-2099)
	IMON	[INTEGER *2]	: MONTH (1=JAN, ETC.)
	IDAY	[INTEGER *2]	: DAY (1 - 31)

;SET UP COMMUNICATION FRAME WITH CALLER

FRAME	STRUC		
GDATE_BP	DW	?	;CALLER'S BP

```

GDATE_RET      DD      ?      ;RETURN ADDRESS
IDAY           DD      ?      ;RETURN CELL FOR DAY
IMON           DD      ?      ;RETURN CELL FOR MONTH
IYR            DD      ?      ;RETURN CELL FOR YEAR
FRAME          ENDS

;
;REQUIRED FOR FORTRAN CALL
CSEG           SEGMENT 'CODE'
DGROUP         GROUP DATA
               ASSUME CS:CSEG,DS:DGROUP,ES:DGROUP,SS:DGROUP
GDATE          PROC FAR
               PUBLIC GDATE
               PUSH BP      ;SAVE CALLER'S BP
               MOV BP,SP    ;SET FRAME BASE
;
;PROCESS DATE REQUEST
               MOV AH,2AH   ;DOS DATE REQUEST
               INT 21H      ;CALL DOS TO GET DATE
               LES BX,[BP]+IYR ;GET IYR ADDRESS
               MOV ES:[BX],CX ;PUT YEAR INTO IYR
               SUB AX,AX     ;PARMS GO INTO AL
               MOV AL,DH     ;GET THE MONTH
               LES BX,[BP]+IMON ;IMON ADDRESS
               MOV ES:[BX],AX ;PUT MONTH INTO IMON
               MOV AL,DL     ;GET THE DAY
               LES BX,[BP]+IDAY ;IDAY ADDRESS
               MOV ES:[BX],AX ;PUT DAY INTO IDAY
               POP BP        ;RESTORE CALLER'S BP
               RET 12        ;FREE 6 PARMS ON RETURN
GDATE          ENDP
CSEG           ENDS
               END

;SOURCE FILE:  GTIME.ASM
;-----
; 01 JUNE 1983  ORVILLE D. GREEN  IBM/PC  DOS 2.0
;      GTIME RETURNS THE CURRENT TIME
;
; CALLING SEQUENCE :  CALL GTIME (IHR,IMIN,ISEC,IHS)
;      IHR [INTEGER *2] : CURRENT HOUR (0-23)
;      IMIN [INTEGER *2] : CURRENT MINUTE (0-59)
;      ISEC [INTEGER *2] : CURRENT SECOND (0-59)
;      IHS [INTEGER *2] : CURRENT 1/100 SECOND (0-99)
;-----
;SET UP COMMUNICATION FRAME WITH CALLER
FRAME          STRUC
GTIME_BP       DW      ?      ;CALLER'S BP
GTIME_RET      DD      ?      ;RETURN ADDRESS
IHS            DD      ?      ;RETURN CELL FOR 1/100 SEC
ISEC           DD      ?      ;RETURN CELL FOR SECONDS
IMIN           DD      ?      ;RETURN CELL FOR MINUTE

```

```

IHR          DD          ?          ;RETURN CELL FOR HOUR
FRAME        ENDS

;
;REQUIRED FOR FORTRAN CALL
CSEG         SEGMENT      'CODE'
DGROUP       GROUP        DATA
            ASSUME CS:CSEG,DS:DGROUP,ES:DGROUP,SS:DGROUP
GTIME        PROC         FAR
            PUBLIC        GTIME
            PUSH          BP          ;SAVE CALLER'S BP
            MOV           BP,SP      ;SET FRAME BASE
;
;PROCESS TIME REQUEST
            MOV           AH,2CH      ;DOS REQUEST TIME FUNCTION
            INT           21H        ;CALL DOS TO GET THE TIME
            SUB           AX,AX      ;PARMS GO INTO AL
            MOV           AL,CH      ;GET HOURS
            LES           BX,[BP]+IHR ;IHR ADDRESS
            MOV           ES:[BX],AX ;PUT HOUR INTO IHR
            MOV           AL,CL      ;GET MINUTES
            LES           BX,[BP]+IMIN ;IMIN ADDRESS
            MOV           ES:[BX],AX ;PUT MINUTE INTO IMIN
            MOV           AL,DH      ;GET SECONDS
            LES           BX,[BP]+ISEC ;ISEC ADDRESS
            MOV           ES:[BX],AX ;PUT SECONDS INTO ISEC
            MOV           AL,DL      ;GET 1/100 SECONDS
            LES           BX,[BP]+IHS ;IHS ADDRESS
            MOV           ES:[BX],AX ;PUT 1/100 SECONDS INTO IHS
            POP           BP         ;RESTORE CALLER'S BP
            RET           16         ;FREE 8 PARMS ON RETURN
GTIME        ENDP
CSEG         ENDS
            END

;SOURCE FILE:  HOME.ASM
;-----
; 01 JUNE 1983  ORVILLE D. GREEN  IBM/PC  DOS 2.0
;   HOME MOVES THE CURSOR TO THE UPPER LEFT HAND CORNER OF THE SCREEN
;
;CALLING SEQUENCE :  CALL HOME
;-----
;
;SET UP COMMUNICATION FRAME WITH CALLER
FRAME        STRUC
HOME_BP      DW           ?          ;CALLER'S BP
HOME_RET     DD           ?          ;RETURN ADDRESS
FRAME        ENDS
;
;REQUIRED FOR FORTRAN CALL
CSEG         SEGMENT      'CODE'
DGROUP       GROUP        DATA

```

```

HOME      ASSUME  CS:CSEG,DS:DGROUP,ES:DGROUP,SS:DGROUP
          PROC    FAR
          PUBLIC  HOME
          PUSH    BP          ;SAVE CALLER'S BP
          MOV     BP,SP      ;SET FRAME BASE
;
;PROCESS HOME REQUEST
          MOV     AH,2        ;BEAM POSITION FUNCTION
          MOV     DX,0        ;SPECIFY POS (0,0)
          MOV     BH,0        ;SCREEN # TO USE
          INT     10H         ;CALL BIOS TO MOVE CURSOR
          POP     BP          ;RESTORE CALLER'S BP
          RET             ;FREE 2 PARMS ON RETURN
HOME      ENDP
CSEG      ENDS
          END

;-----LOOK-----
; 23 APRIL 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
; LOOK PERFORMS A DIRECTORY SEARCH FOR THE INDICATED FILE, USING
; DOS INTERRUPT 21H, FUNCTION 11H
;ENTRY
; FCB (37 BYTES) UNOPENED STANDARDFILE CONTROL BLOCK (SEE BELOW)
; DRIVE (INTEGER) DISK DRIVE NUMBER
;EXIT
; FOUND (INTEGER) = 0 FILE FOUND
;                  = 255 FILE NOT FOUND
; FCB MODIFIED (SEE DOS MANUAL)
;
;****INTEGER VALUES MAY BE 2 OR 4 BYTES
;
;FORTHAN CALLING SEQUENCE
; CALL LOOK (FCB,DRIVE,FOUND)
;NOTE: ENCODE FCB WITH :
; BYTE 1 BLANK
; BYTES 2 - 9 FILE NAME, LEFT-JUSTIFIED, TRAILING BLANKS
; BYTES 10 - 12 FILE EXTENSION, LEFT-JUSTIFIED, TRAILING BLANKS
; (MAY BE ALL BLANK -- NO DOT)
; BYTES 13 - 37 BLANK
;
;*****NO ERROR TRAP INITIATED FOR "DRIVE NOT READY" ERRORS
;-----
;COMMUNICATION FRAME WITH CALLER
FRAME     STRUC
LOOK_BP   DW      ?          ;CALLER'S BP
LOOK_RET  DD      ?          ;RETURN ADDRESS
FOUND     DD      ?          ;FOUND ADDRESS
DRIVE     DD      ?          ;DRIVE ADDRESS
FCB       DD      ?          ;FCB ADDRESS
FRAME     ENDS
;

```

```

;REQUIRED FOR FORTRAN CALL
CSEG      SEGMENT      'CODE'
DGROUP    GROUP        DATA
          ASSUME CS:CSEG,DS:DGROUP,ES:DGROUP,SS:DGROUP
LOOK      PROC          FAR
          PUBLIC        LOOK
          PUSH          BP                      ;SAVE BP
          MOV           BP,SP                  ;SET FRAME BASE
;PERFORM SEARCH VIA DOS INTERRUPT 21H, FUNCTION 11H
          LES           BX,[BP]+DRIVE          ;DRIVE NUMBER ADDRESS
          MOV           AX,ES:[BX]            ;DRIVE # IN AX
          LES           BX,[BP]+FCB           ;FCB STARTING ADDRESS
          MOV           DX,BX                 ;DS:DX POINT TO FCB
          MOV           DS:[BX],AL           ;PUT DRIVE # INTO FCB
          MOV           AH,11H                ;DOS FIRST FILE REQUEST
          INT           21H                  ;CALL DOS TO DO SEARCH
;PUT FOUND CODE IN RETURN AREA AND QUIT
          LES           BX,[BP]+FOUND          ;FOUND ADDRESS
          MOV           ES:[BX],AL           ;(AL) HAS FOUND
          POP           BP                    ;RESTORE BP
          RET           8                     ;FAR RETURN
LOOK      ENDP
CSEG      ENDS
          END

;-----REMOVE-----
; 15 MAY 1984 ORVILLE D. GREEN IBM/PC DOS 2.0
; REMOVE DELETES A FILE FROM THE DIRECTORY, USING DOS INTERRUPT
; 21H, FUNCTION 11H
;ENTRY
; FCB (37 BYTES) UNOPENED STANDARDFILE CONTROL BLOCK (SEE BELOW)
; DRIVE (INTEGER) DISK DRIVE NUMBER
;EXIT
; STATUS (INTEGER) = 0 DELETE SUCCESSFUL
;                  = 255 DELETE NOT SUCCESSFUL (FILE NOT FOUND)
; FCB MODIFIED (SEE DOS MANUAL)
;
;FORTRAN CALLING SEQUENCE
; CALL REMOVE (FCB,DRIVE,STATUS)
;NOTE: ENCODE FCB WITH :
; BYTE 1          BLANK
; BYTES 2 - 9     FILE NAME, LEFT-JUSTIFIED, TRAILING BLANKS
; BYTES 10 - 12   FILE EXTENSION, LEFT-JUSTIFIED, TRAILING BLANKS
;                 (MAY BE ALL BLANK -- NO DOT)
; BYTES 13 - 37   BLANK
;
;*****NO ERROR TRAP INITIATED FOR "DRIVE NOT READY" ERRORS
;-----
;COMMUNICATION FRAME WITH CALLER
FRAME      STRUC
REMOVE_BP  DW          ?                      ;CALLER'S BP

```

```

REMOVE_RET    DD      ?      ;RETURN ADDRESS
STATUS        DD      ?      ;STATUS ADDRESS
DRIVE         DD      ?      ;DRIVE ADDRESS
FCB           DD      ?      ;FCB ADDRESS
FRAME        ENDS

;
;REQUIRED FOR FORTRAN CALL
CSEG          SEGMENT      'CODE'
DGROUP        GROUP        DATA
              ASSUME CS:CSEG,DS:DGROUP,ES:DGROUP,SS:DGROUP
REMOVE        PROC          FAR
              PUBLIC        REMOVE
              PUSH          BP      ;SAVE BP
              MOV           BP,SP   ;SET FRAME BASE
;DELETE FILE VIA DOS INTERRUPT 21H, FUNCTION 13H
              LES           BX,[BP]+DRIVE ;DRIVE NUMBER ADDRESS
              MOV           AX,ES:[BX] ;DRIVE # IN AX
              LES           BX,[BP]+FCB ;FCB STARTING ADDRESS
              MOV           DX,BX     ;DS:DX POINT TO FCB
              MOV           DS:[BX],AL ;PUT DRIVE # INTO FCB
              MOV           AH,13H    ;DOS DELETE FILE REQUEST
              INT           21H       ;CALL DOS TO DO DELETE
;PUT STATUS CODE IN RETURN AREA AND QUIT
              LES           BX,[BP]+STATUS ;STATUS ADDRESS
              MOV           ES:[BX],AL    ;(AL) HAS STATUS
              POP           BP           ;RESTORE BP
              RET           8           ;FAR RETURN
REMOVE        ENDP
CSEG          ENDS
              END

```

Rawinsonde INCLUDE Files

*RWSTN.DCL -- RAWIN station data area

```

      INTEGER      STNMAX,STNNUM
      REAL          STLATX(3,6),STLONX(3,6),STHGT(6)

      COMMON /RWSTN/  STNMAX,STNNUM,STLATX,STLONX,STHGT

      CHARACTER     STID12(6)*12,STID04(6)*4,STID01(6)*1

      COMMON /RWSTNC/ STID12,STID04,STID01

```

*RWIDX.DCL -- RAWIN index file data area

```

      INTEGER      IDXLU,IDXDX,IDXMAX,IDXLEN,IDXDAT(25),IDXMIN(25,6)
      INTEGER      IDXNUM,BEGDAT,ENDDAT,MSGSD,GETNUM
      INTEGER      GETMAX,GETLEN,GETDAT(50),GETSTN(50),GETMIN(50,6)

      COMMON /RWIDX/  IDXLU,IDXDX,IDXMAX,IDXLEN,IDXDAT,IDXMIN
      COMMON /RWIDX/  IDXNUM,BEGDAT,ENDDAT,MSGSD,GETNUM
      COMMON /RWIDX/  GETMAX,GETLEN,GETDAT,GETSTN,GETMIN

      CHARACTER     IDXNAM*12,IDXFIL*14

      COMMON /RWIDXC/  IDXNAM,IDXFIL

```

*RWMAP.DCL -- RAWIN map data

```

      REAL          LATR,LONR,LATF,LONF,LAT(6),LON(6)
      REAL          XF,YF,XS(6),YS(6),XDX(200,6),YDX(200,6)

      COMMON /RWMAP/  LATR,LONR,LATF,LONF,LAT,LON
      COMMON /RWMAP/  XF,YF,XS,YS,XDX,YDX

```

\$INCLUDE:'RAWINTSK.DCL'

\$INCLUDE:'RWINT.DCL'

\$INCLUDE:'RWTBL.DCL'

\$INCLUDE:'RWDTA.DCL'

\$INCLUDE:'RWMAP.DCL'

*RWJOB.DCL -- RAWIN jobfile data area

```

      INTEGER      JOBLU,JOBDX

      COMMON /RWJOB/  JOBLU,JOBDX

      CHARACTER     JOBNAM*12,JOBFIL*14,JOBDAT*22

      COMMON /RWJOBC/  JOBNAM,JOBFIL,JOBDAT

```

```

DATA PMX /0.0,20000.0,20000.0, 60000.0, 60000.0, 0.0,
-        30.0, 30.0, 350.0, 350.0, 450.0,
-        50.0, 360.0, 50.0, 50.0,
-        20.0, 20.0, 0.0015, 0.0015, 100.0,
-        0.0006, 0.0006, 0.0006, 0.0006, 95.0,
-        0.0, 0.0, 0.0, 0.0, 2.0/

```

```

DATA PMN /1000.0, 0.0, 0.0, 0.0, 0.0, 60.0,
-        -30.0, -30.0, 250.0, 250.0, 250.0,
-        0.0, 0.0, 0.0, 0.0,
-        0.0, 0.0, 0.0, 0.0, 0.0,
-        -0.0006, -0.0006, -0.0006, -0.0006, 70.0,
-        0.0, 0.0, 0.0, 0.0, -2.0/

```

*RWMSC.DCL -- RAWIN miscellaneous data area

```

INTEGER          CLR,SET,BIT0,BIT1,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7

```

```

COMMON /RWMSC/   CLR,SET,BIT0,BIT1,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7

```

```

CHARACTER        DRIVE(9)*2,MONTH(12)*3

```

```

COMMON /RWMSCC/  DRIVE,MONTH

```

*RAWINGET.DCL -- RAWINGET common inserts

```

$INCLUDE:'RWMSC.DCL'

```

```

$INCLUDE:'RWIDX.DCL'

```

```

$INCLUDE:'RWSTN.DCL'

```

```

$INCLUDE:'RWSND.DCL'

```

*RWTBL.DCL -- RAWIN table data area

```

INTEGER          TBLLU,TBLDX,TBLMAX,TBLLEN,TBLLIN

```

```

REAL             TBLVAL(400,7)

```

```

COMMON /RWTBL/   TBLLU,TBLDX,TBLMAX,TBLLEN,TBLLIN,TBLVAL

```

```

CHARACTER        TBLNAM*12,TBLFIL*14

```

```

COMMON /RWTBLC/  TBLNAM,TBLFIL

```

*RAWINTSK.DCL -- RAWIN task common

```

$INCLUDE:'RWMSC.DCL'

```

```

$INCLUDE:'RWIDX.DCL'

```

```

$INCLUDE:'RWSTN.DCL'

```

```

$INCLUDE:'RWSND.DCL'

```

```

$INCLUDE: 'RWPOC.DCL'
$INCLUDE: 'RWSFC.DCL'
$INCLUDE: 'RWTSC.DCL'
$INCLUDE: 'RWDMC.DCL'
$INCLUDE: 'RWPFC.DCL'
$INCLUDE: 'RWPMC.DCL'

```

```

$INCLUDE: 'RWTSK.DCL'
$INCLUDE: 'RWFLG.DCL'

```

*RWTSK.DCL -- RAWIN task description data area

```

INTEGER      TSKLU,TSKDX
INTEGER      DMNUM,TSNUM,STNREQ,TTINT
INTEGER      TBHMAX,TBHSPC(6),TBHLEN,TBHNUM
INTEGER      TBCMAX,TBCSPC(6),TBCLEN,TBCNUM
INTEGER      TBYSPC,PFCSPC(6)
REAL         TBYINC

```

```

COMMON /RWTSK/ TSKLU,TSKDX
COMMON /RWTSK/ DMNUM,TSNUM,STNREQ,TTINT
COMMON /RWTSK/ TBHMAX,TBHSPC,TBHLEN,TBHNUM
COMMON /RWTSK/ TBCMAX,TBCSPC,TBCLEN,TBCNUM
COMMON /RWTSK/ TBYSPC,TBYINC,PFCSPC

```

```

CHARACTER    TSKNAM*12,TSKFIL*14
COMMON /RWTSKC/ TSKNAM,TSKFIL

```

*RWSFC.DCL -- RAWIN processing flag codes

```

INTEGER      SFLEN,SFBIT(6)
INTEGER      BD,PS,SV,IO,FD,AS

```

```

COMMON /RWSFC/ SFLEN,SFBIT
COMMON /RWSFC/ BD,PS,SV,IO,FD,AS

```

```

INTEGER      SFOPCN(6)
EQUIVALENCE  (SFOPCN(1),BD)

```

```

CHARACTER    SFOPC(6)*2,SFDES(6)*30,SFTTL*30

```

```

COMMON /RWSFCC/ SFOPC,SFDES,SFTTL

```

*RWFLG.DCL -- RAWIN processing flags data area

```

LOGICAL      BDFLG,PSFLG,SVFLG,IOFLG,FDFLG,ASFLG
LOGICAL      MSFLG,APFLG

```

```

COMMON /RWFLG/ BDFLG,PSFLG,SVFLG,IOFLG,FDFLG,ASFLG
COMMON /RWFLG/ MSFLG,APFLG

```

LOGICAL RWFLGX(1)
EQUIVALENCE (RWFLGX(1),BDFLG)

*RWPFCL DCL -- RAWIN plot format codes

INTEGER PFLN,PFBIT(2)
INTEGER IX,MX

COMMON /RWPFCL/ PFLN,PFBIT
COMMON /RWPFCL/ IX,MX

INTEGER PFOPCN(2)
EQUIVALENCE (PFOPCN(1),IX)

CHARACTER PFOPC(2)*2,PFDES(2)*30,PFTTL*30

COMMON /RWPFCC/ PFOPC,PFDES,PFTTL

*RWPOCL DCL -- RAWIN processing option codes

INTEGER POLEN,POBIT(12)
INTEGER NP,SF,SP,SX,DM,TS,PF,DF,DT,WT,TI,EN

COMMON /RWPOCL/ POLEN,POBIT
COMMON /RWPOCL/ NP,SF,SP,SX,DM,TS,PF,DF,DT,WT,TI,EN

INTEGER POOPCN(12)
EQUIVALENCE (POOPCN(1),NP)

CHARACTER POOPC(12)*2,PODES(12)*30,POTTTL*30

COMMON /RWPOCC/ POOPC,PODES,POTTTL

*RWPMCL DCL -- RAWIN parameter codes

INTEGER PMLN,PMBIT(30)
INTEGER PR,HM,ZM,HF,ZF,TM,TE,TD,TV,TP
INTEGER TQ,WS,WD,UU,VV,EA,ES,MR,SH,RH
INTEGER CV,RV,AV,MC,AE,PE,KE,XD,YD,WW

COMMON /RWPMCL/ PMLN,PMBIT
COMMON /RWPMCL/ PR,HM,ZM,HF,ZF,TM,TE,TD,TV,TP
COMMON /RWPMCL/ TQ,WS,WD,UU,VV,EA,ES,MR,SH,RH
COMMON /RWPMCL/ CV,RV,AV,MC,AE,PE,KE,XD,YD,WW

INTEGER PMOPCN(30)
EQUIVALENCE (PMOPCN(1),PR)

CHARACTER PMOPC(30)*2,PMDES(30)*30,PMTTL*30

CHARACTER PMCOL(30)*12,PMFMT(30)*12

COMMON /RWPMCC/ PMOPC,PMDES,PMTTL,PMCOL,PMFMT

*RWPLT.DCL -- RAWIN plot data

REAL PMMN(30),PMMX(30)

INTEGER IPXY(44)

COMMON /RWPLT/ PMMN,PMMX,IPXY

*RWTSC.DCL -- RAWIN task specification codes

INTEGER TSLEN,TSBIT(3)

INTEGER MP,IP,PP

COMMON /RWTSC/ TSLEN,TSBIT

COMMON /RWTSC/ MP,IP,PP

INTEGER TSOPCN(3)

EQUIVALENCE (TSOPCN(1),MP)

CHARACTER TSOPC(3)*2,TSDES(3)*30,TSTTL*30

COMMON /RWTSCC/ TSOPC,TSDES,TSTTL

*RWDTA.DCL -- RAWIN job data area

REAL PRX(200,6),TEX(200,6),TDX(200,6)

REAL WSX(200,6),WDX(200,6),TMX(200,6)

REAL HMX(200,6),YYX(200,6)

INTEGER NX(6)

COMMON /RWDTA/ NX,PRX,TEX,TDX,WSX,WDX,TMX,HMX,YYX

REAL PDX(9600)

EQUIVALENCE (PDX(1),PRX(1,1))

*RWINT.DCL -- RAWIN interpolated data

REAL PRI(6),TEI(6),TDI(6),WSI(6),WDI(6),TMI(6)

REAL HMI(6),UUI(6),VVI(6),XDI(6),YDI(6)

REAL PRI0(6),TEI0(6),TDI0(6),WSI0(6),WDI0(6),TMI0(6)

REAL HMI0(6),UUI0(6),VVI0(6),XDI0(6),YDI0(6)

COMMON /RWINT/ PRI,TEI,TDI,WSI,WDI,TMI,HMI,UUI,VVI,XDI,YDI

COMMON /RWINT/ PRI0,TEI0,TDI0,WSI0,WDI0,TMI0,HMI0,UUI0,VVI0

COMMON /RWINT/ XDI0,YDI0

REAL PDI(66)

EQUIVALENCE (PDI(1),PRI(1))

*RWDMC.DCL -- RAWIN display mode codes

INTEGER DMLen,DMBit(3)
 INTEGER TB,PL,BO

COMMON /RWDMC/ DMLen,DMBit
 COMMON /RWDMC/ TB,PL,BO

INTEGER DMOPCN(3)
 EQUIVALENCE (DMOPCN(1),TB)

CHARACTER DMOPC(3)*2,DMDES(3)*30,DMTTL*30

COMMON /RWDMCC/ DMOPC,DMDES,DMTTL

*RWSND.DCL -- RAWIN sounding file data area

INTEGER SNDLU,SNDDX,SNDPRM,SNDMAX,SNDLEN,SNDRPT
 INTEGER SNDLDP,SNDDMT,SNDMIN,SNDGMT,MDTGMT

REAL SNDPRL,SNDMSG,SNDLAT(3),SNDLON(3),SNDHGT
 REAL SNDDTA(200,6)

REAL PRC(200),TEC(200),TDC(200)
 REAL WSC(200),WDC(200),TMC(200)
 EQUIVALENCE (PRC(1),SNDDTA(1,1)), (TEC(1),SNDDTA(1,2))
 EQUIVALENCE (TDC(1),SNDDTA(1,3)), (WSC(1),SNDDTA(1,4))
 EQUIVALENCE (WDC(1),SNDDTA(1,5)), (TMC(1),SNDDTA(1,6))

COMMON /RWSND/ SNDLU,SNDDX,SNDPRM,SNDMAX,SNDLEN,SNDRPT
 COMMON /RWSND/ SNDLDP,SNDDMT,SNDMIN,SNDGMT,MDTGMT
 COMMON /RWSND/ SNDPRL,SNDMSG,SNDLAT,SNDLON,SNDHGT,SNDDTA

CHARACTER SNDNAM*12,SNDFIL*14,SNDSTN*12

COMMON /RWSNDC/ SNDNAM,SNDFIL,SNDSTN

APPENDIX B

LSA PROGRAM LISTINGS

APPENDIX B

LSA PROGRAM LISTINGS

LSA Front-endPMKIN mainline

\$NOFLOATCALLS

PROGRAM PMKIN

C...MAINLINE FOR POOR MAN'S KENEMATICS.

C...WRITTEN FALL 1983, J. A. HEIMBACH, JR.

C...840911: PRECIP OPTION CHANGED FROM AVERAGING OVER TIME TO CUMULATIVE $\frac{1}{2}$

C...840911: WIND AVGS FOR MULTIPLE STATIONS CORRECTED. U, V AND FFF

C WERE O.K.

C...840914: (ODG) PMKIN SYSTEM SEGMENTED TO TAKE ADVANTAGE OF MS-DOS

C 3.20 FORTRAN, AND TO REDUCE THE SIZE OF THE IMAGE CODE THAT

C IS MEMORY-RESIDENT DURING EXECUTION. SEGMENTATION IS AS

C FOLLOWS:

C 1. PMKIN - DOES NOT CALL XY3PLT TO PRODUCE PLOTS, RATHER
C WRITES ARGUMENTS TO BINARY FILES (\$PMKIN.nnn)C 2. PMKINPLT - PMKIN BACKEND PROGRAM. READS \$PMKIN.nnn
C FILES, SCHEDULES XY3PLT.

C 3. XY3PRT - PRODUCES HARDCOPIES OF PLOTFILES FROM 2.

C PMOUT SUBROUTINE ADDED TO BUILD \$PMKIN.nnn FILES:

CALLS PMOUT

C...850617 - EXTERNAL SEGMENT (PMKPP) ADDED TO CLIP PLOTS

C ***CAUTION*** DIMENSION CHANGES TO PMOUT ARGUMENTS REQUIRE
C CORRESPONDING CHANGES TO PMKINPLT AND PMOUT
C DIMENSIONS.

C...LOGICAL UNITS: * = CON = DEFAULT,

C 1 = MISC DATA INPUT, DEFAULT = CON: ,

C 6 = DUMP OF PARAMETERS.

C (MESONET DATA SCESS HANDLED BY TOM'S SOFTWARE.

C

C...ENGLE'S LIBRARY ITEMS NEEDED (CONTAINED IN \$XY3PLT.FOR):

C INPUT MDATA RDATA MLOCS OPENMS

C READMS STINDX CLOSMS BADFPT SYDYS

C MSUCHK LEGVAR C2I2C DELTAP

CALLS INPUT

CALLS MDATA

CALLS RDATA

CALLS MLOCS

CALLS OPENMS

CALLS READMS

CALLS STINDX

CALLS CLOSMS
 CALLS BADFPT
 CALLS SYDYMS
 CALLS MSUCHK
 CALLS LEGVAR
 CALLS C2I2C
 CALLS DELTAP

C...J. A. H.'S ROUTINES NEEDED (CONTAINED IN PMKINAD.FOR):

C	PAREA	PCON	NEXT	LAMB	TITLES
C	PVORT	AVG	HUM	POT	PSEA
C	PMCON	RMWRK	TOTAL		

CALLS PAREA
 CALLS PCON
 CALLS NEXT
 CALLS LAMB
 CALLS TITLES
 CALLS PVORT
 CALLS AVG
 CALLS HUM
 CALLS POT
 CALLS PSEA
 CALLS PMCON
 CALLS RMWRK
 CALLS TOTAL
 C

INTEGER SDATE,STIME,EDATE,ETIME,DATE,TIME,ERR,ANAL,AT
 CHARACTER*1 BELL,BS,DDRIVE,RDRIVE,YES,NO,Q
 CHARACTER STN(50)*3,OUT*10,OPT(20)*40
 REAL LATR,LONGR,LATF,LONGF,MSNG
 DIMENSION ZVAR(6),X(50),Y(50),XPLT(865),YPLT(865),ZELE(50)
 DIMENSION PR(50),TE(50),RH(50),UU(50),VV(50),PC(50),TEMP(50)
 DIMENSION DELP(50,2)
 LOGICAL PROMPT,RECAL,DUMP,NMSG

C...A QUICK NOTE ON RUNNING MEAN STASHING. RM(L,M,N): L = STATION INDEX,
 C...M = PARAMETER WHERE 1 THROUGH 6 ARE PR, TE, RH, UU, VV RESPECTIVELY.
 C...PC IS NOT INCLUDED IN RUNNING MEAN PROCESS. N USED TO INDEX 5-MIN PERIOD.

COMMON /RMAVG/ RM(50,5,12),AV(5)

C\$\$

C....ODG ADDITION (SUPPORTING SOFTWARE FOR BACKEND IN CONTAINED IN
 C...XY3PLT.FOR AND &XY3PLT.FOR):

LOGICAL PLOT
 CHARACTER TX(3,15)*4,LF*14,AVGTIM*20,TSPAN*48,NUMSTA*12
 CHARACTER DSKDRV(10)*1,CLD*1,MISC*14
 REAL PMIN(17),PMAX(17)
 INTEGER IP(10)
 DATA DSKDRV /'A','B','C','D','E','F','G','H','I','J'/
 DATA PMIN /850.,10.,0.,-50.,-50.,-10.,950.,250.,250.,0.0,
 1 0.,0.,-0.000999,-0.000999,-0.000999,-1.0,75./
 DATA PMAX /950.,40.,1.,50.,50.,50.,1050.,350.,350.,0.001,

```

1          360.,50.,0.000999,0.000999,0.0000999,1.0,85./
DATA LF/' ',ANAL/1/
DATA IP(1),IP(2),IP(3),IP(4),IP(5),IP(6),IP(7),IP(8),IP(9),IP(10)
*/0,0,0,4,0,0,0,0,0,0/
C$$
DATA YES,NO/'Y','N'/
DATA LATR,LONGR/46.4281,105.8733/,LATF,LONGF/45.75,105.90/
DATA MSNG/-1.2345E+38/
CALL TITLES(OPT,NOPT)
BELL=CHAR(7)
BS=CHAR(8)
PROMPT=.FALSE.
100  FORMAT(1X,A)
101  FORMAT(1X,A,*)
102  FORMAT(A)
103  FORMAT(1X,7A*)
104  FORMAT(1X,5A*)
105  FORMAT(1X,17A*)
106  FORMAT(F7.4,1X,F8.4)
107  FORMAT(1X,A,A)
108  FORMAT(19X,I2,A,A)
WRITE(*,100) ' POOR MAN'S KINEMATICS AT YOUR SERVICE.'
*,
WRITE(*,101) ' SOURCE FOR MISC. INPUT (CR=CON:)? '
READ(*,102) MISC
IF(MISC.EQ.' ') THEN
    OPEN(1,FILE='CON:')
ELSE
    OPEN(1,FILE=MISC)
ENDIF
WRITE(*,101) ' WHICH DRIVE CONTAINS MESONET DATA? '
READ(1,102) DDRIVE
WRITE(*,101) 'AND WHICH DRIVE CONTAINS RECALIBRATION DATA? '
READ(1,102) RDRIVE
WRITE(*,101) 'WHERE IS PRINTED OUTPUT TO BE DIRECTED? '
READ(1,102) OUT
OPEN(6,FILE=OUT,STATUS='NEW')
218 WRITE(*,103) 'START DATE?  '  ' ',BS,BS,BS,BS,BS,BS
READ(1,'(I6)') SDATE
WRITE(*,104) 'START TIME?  '  ' ',BS,BS,BS,BS
READ(1,'(I4)') STIME
WRITE(*,103) 'END DATE?  '  ' ',BS,BS,BS,BS,BS,BS
READ(1,'(I6)') EDATE
WRITE(*,104) 'END TIME (INCL.)?  '  ' ',BS,BS,BS,BS
READ(1,'(I4)') ETIME
WRITE(*,101) 'DO YOU WANT RECALIBRATIONS APPLIED (Y/N)? '
READ(1,102) Q
RECAL=Q.EQ.'Y'
C...WORK ON MAP SPECIFICATIONS.
WRITE(*,101) 'DO YOU WANT THE DEFAULT MAP SPECIFICATIONS (Y/N)? '

```

```

      READ(1,102) Q
      IF(Q.EQ.'Y') GO TO 3
      WRITE(*,105) 'INPUT REFERENCE LAT., LONG. (DD.DDDD DDD.DDDD) '
*      ' ' ' ' ,BELL,(BS,I=1,16)
      READ(1,106) LATR, LONGR
      WRITE(*,105) 'INPUT LAT., LONG. OF SW CORNER (DD.DDDD DDD.DDDD) '
*      ' ' ' ' , (BS,I=1,16)
      READ(1,106) LATF, LONGF
3      WRITE(*,101) 'DO YOU WANT INFORMATIONAL AND DEBUG ITEMS DUMPED (Y/
*N)? '
      READ(1,102) Q
      DUMP=Q.EQ.'Y'

C$$
C. . .ODG ADDITION
      WRITE(*,101) 'DO YOU WANNA PLOT IT ? (Y/N) '
      READ(1,102) Q
      PLOT = Q .EQ. 'Y'
      IF(OPT(ANAL).EQ.'START FROM BEGINNING') GO TO 20
      IF (PLOT) THEN
        XSN = 0.0
        WRITE(*,101) 'DRIVE FOR LOG/PLOT FILE ? '
        READ(1,'(A1)') CLD
        DO 88 I = 1, 10
          IF(CLD .EQ. DSKDRV(I)) GOTO 89
88      CONTINUE
          I = 0
89      LD = I
          WRITE(*,101) 'PAUSE FOR LOG CREATE (Y/N) ? '
          READ(1,'(A)') Q
          IF (Q .EQ. 'Y') LP = 0
          IF (Q .NE. 'Y') LP = 1
          LU = 57
        ENDIF

C$$
20      WRITE(6,'(A,F7.4,1X,F8.4,/,A,F7.4,1X,F8.4)')
*      ' REF. LAT. AND LONG. = ',LATR, LONGR
*,      ' FIDUCIAL LAT. AND LONG. = ',LATF, LONGF
      CALL LAMB(LATR, LONGR, LATF, LONGF, ZLAT, ZLONG, XF, YF, X(1), Y(1), .TRUE.)
      WRITE(6,'(A,F8.0,A,/,24X,A,2X,F8.0,A)')
*      ' FIDUCIAL POINT (SW CORNER) = ',YF, ' METERS SOUTH OF NORTH POLE'
*,      'AND ',XF, ' METERS WEST OF REF. LONG.'
      WRITE(*,'(/,1X,4A@)') 'INPUT NUMBER OF MINUTES TO AVERAGE ' ' '
*,BS,BS
      READ(1,'(I2)') AT
C...BE SURE DUMMIES GET AT LEAST 5 MIN.
      IF(AT.LT.5) AT=5
C...ATH = DECIMAT DELTA-T. GO BACK ONE HALF DELTA-T.
      ATH=FLOAT(5)/60.
C...FIND INITIAL CENTERED TIME. REMEMBER THAT REPORT TIME REPRESENTS

```

C...THE AVERAGE OF THE PREVIOUS 5 MIN.

```
219  T=INT(STIME/100.)
      ST=T+((REAL(STIME)-(T*100.))/60)-(FLOAT(AT)/120.)-ATH
      NRM=((FLOAT(AT)+2.5)/5.0)+0.0001
```

C...SCARF UP STATIONS AND FIND THEIR POSITIONS.

```
      WRITE(*,107) 'INPUT STATION ID'S FOR ANALYSIS, E.G., MLS, ONE PER
*LINE.  BLANK = DONE.'
```

```
      NSTN=0
```

```
1    NSTN=NSTN+1
      WRITE(*,'(1X,4A,*)') 'I',BS,BS,BS
      READ(1,102) STN(NSTN)
      IF(STN(NSTN).NE.' ') GO TO 1
      NSTN=NSTN-1
      WRITE(6,'(/,A/,A)')
```

```
* '  I    STN    LAT    LONG    ELEV    X(M)    Y(M)  '
*, ' -----'
```

```
DO 2 I=1,NSTN
```

```
  CALL MLOCS(STN(I),ZLAT,ZLONG,ZELE(I),DDRIVE,PROMPT,ERR)
```

C...USE WEST LONG AS POS. TOM SET UP MESOLOCS.--- AS NEG.

```
  ZLONG=-ZLONG
```

```
  CALL LAMB(LATR,LONGR,LATF,LONGF,ZLAT,ZLONG,XF,YF,X(I),Y(I),.FALSE.
*)
```

```
  WRITE(6,'(1X,I3,3X,A,2F10.4,2X,F5.0,2X,2F10.0)') I,STN(I)
*,ZLAT,ZLONG,ZELE(I),X(I),Y(I)
```

```
2    CONTINUE
```

```
10   WRITE(*,104) 'ANALYSIS OPTIONS: ', '1. ',OPT(1)
      WRITE(*,108) (I,'. ',OPT(I),I=2,NOPT)
      WRITE(*,'(5A,*)') BELL,BELL,BELL,BELL,BELL
      WRITE(*,'(3A*)') ' MAKE YOUR SELECTION I I',BS,BS
      READ(1,'(I2)') ANAL
      IF(OPT(ANAL).EQ.'QUIT') GO TO 220
      IF(OPT(ANAL).EQ.'NEW STATIONS ONLY') GO TO 219
      IF(OPT(ANAL).EQ.'START FROM BEGINNING') GO TO 218
      IF(ANAL.GT.NOPT.OR.ANAL.LE.0) THEN
        WRITE(*,100) 'DUMMIE! ERRONEOUS OPTION SELECTED.  TRY AGAIN!'
        GO TO 10
```

```
      ENDIF
```

C...INITIALIZE SOME STASH ARRAYS TO MISSING FOR HOUSEKEEPING.

```
DO 19 L=1,50
  DELP(L,1)=MSNG
DO 19 M=1,5
  DO 19 N=1,12
    RM(L,M,N)=MSNG
```

```
19   CONTINUE
      PCUM=0
      DATE=SDATE
      TIME=STIME
      T=ST
      IT=0
8    IT=IT+1
```

```

C...SCARF DATA FOR 5-MIN PERIOD, THEN FIND RUNNING MEANS.
  DO 9 I=1,NSTN
    CALL MDATA(STN(I),DATE,TIME,ZVAR,DDRIVE,PROMPT,ERR)
    IF(DUMP) WRITE(*,'(1X,A,I7,I5,6E10.3,I2)') 'RAW:',DATE,TIME,ZVAR
    *,ERR
C...RECALIBRATE DATA.
  IF(RECAL) THEN
    CALL RDATA(STN(I),DATE,TIME,ZVAR,RDRIVE,PROMPT,ERR)
    IF(DUMP) WRITE(*,'(1X,A,I7,I5,6E10.3,I2)') 'CAL.:',DATE
    *,TIME,ZVAR,ERR
  ENDIF
  RM(I,1,NRM)=ZVAR(1)
  RM(I,2,NRM)=ZVAR(2)
  RM(I,3,NRM)=ZVAR(3)
  RM(I,4,NRM)=ZVAR(4)
  RM(I,5,NRM)=ZVAR(5)
  DELP(I,1)=RM(I,1,1)
  DELP(I,2)=ZVAR(1)
  CALL RMWRK(I,NRM,MSNG)
  PR(I)=AV(1)
  TE(I)=AV(2)
  RH(I)=AV(3)
  UU(I)=AV(4)
  VV(I)=AV(5)
  PC(I)=ZVAR(6)
9  CONTINUE
  GO TO (201,202,203,204,205,206,207,208,209,210,211,212,213
    *,214,215,216,217,220,220,220),ANAL
C...AVERAGE STATION PRESSURE.
201  YPLT(IT)=AVG(NSTN,PR,MSNG)
      GO TO 12
C...AVERAGE TEMPERATURES.
202  YPLT(IT)=AVG(NSTN,TE,MSNG)
      GO TO 12
C...AVERAGE HUMIDITY.
203  YPLT(IT)=AVG(NSTN,RH,MSNG)
      GO TO 12
C...AVERAGE U.
204  YPLT(IT)=AVG(NSTN,UU,MSNG)
      GO TO 12
C...AVERAGE VV.
205  YPLT(IT)=AVG(NSTN,VV,MSNG)
      GO TO 12
C...ACCUMULATE PRECIPITATION. DEFINE ZERO AS THAT PRECIP FOR (1).
C...REMEMBER THAT MDATA RETURNS A DELTA PRECIP.
206  YPLT(IT)=AVG(NSTN,PC,MSNG)
      IF(YPLT(IT).NE.MSNG) THEN
        PCUM=PCUM+YPLT(IT)
        YPLT(IT)=PCUM
      ENDIF

```

```

      GO TO 12
C...SEA LEVEL PRESSURE.
207  DO 13 I=1,NSTN
      TEMP(I)=PSEA(ZELE(I),TE(I),PR(I),MSNG)
13   CONTINUE
      YPLT(IT)=AVG(NSTN,TEMP,MSNG)
      GO TO 12
C...POTENTIAL TEMPERATURE.
208  DO 14 I=1,NSTN
      CALL POT(TE(I),RH(I),PR(I),MSNG,TH,THE)
      TEMP(I)=TH
14   CONTINUE
      YPLT(IT)=AVG(NSTN,TEMP,MSNG)
      GO TO 12
C...EQUIVALENT POTENTIAL TEMPERATURE.
209  DO 15 I=1,NSTN
      CALL POT(TE(I),RH(I),PR(I),MSNG,TH,THE)
      TEMP(I)=THE
15   CONTINUE
      YPLT(IT)=AVG(NSTN,TEMP,MSNG)
      GO TO 12
C...SPECIFIC HUMIDITY.
210  DO 16 I=1,NSTN
      CALL HUM(TE(I),RH(I),PR(I),MSNG,TEMP(I),DUM)
16   CONTINUE
      YPLT(IT)=AVG(NSTN,TEMP,MSNG)
      GO TO 12
C...WIND DIRECTION. GET AVERAGE BY SUMMING UU'S AND VV'S.
211  TEMP(1)=0
      TEMP(2)=0
      NMSNG=.FALSE.
      DO 17 I=1,NSTN
      IF(UU(I).NE.MSNG.AND.VV(I).NE.MSNG) THEN
          TEMP(1)=TEMP(1)+UU(I)
          TEMP(2)=TEMP(2)+VV(I)
          NMSNG=.TRUE.
      ENDIF
17   CONTINUE
      IF(.NOT.NMSNG) TEMP(1)=MSNG
      CALL WIND(TEMP(1),TEMP(2),MSNG,YPLT(IT),DUM)
      GO TO 12
C...WIND SPEED.
212  DO 18 I=1,NSTN
      CALL WIND(UU(I),VV(I),MSNG,DUM,TEMP(I))
18   CONTINUE
      YPLT(IT)=AVG(NSTN,TEMP,MSNG)
      GO TO 12
C...KINEMATIC CONVERGENCE.
213  CALL PCON(NSTN,X,Y,UU,VV,MSNG,YPLT(IT))
      GO TO 12

```

```

C...VORTICITY.
214  CALL PVORT(NSTN,X,Y,UU,VV,MSNG,YPLT(IT))
      GO TO 12
C...MOISTURE CONVERGENCE.
215  CALL PMCON(NSTN,X,Y,PR,TE,RH,UU,VV,MSNG,YPLT(IT))
      GO TO 12
C...PRESSURE TREND.
216  CALL DELTAP(DELP,NSTN,MSNG,YPLT(IT))
      GO TO 12
C...TOTAL ATMOSPHERIC ENERGY.
217  DO 2171 I=1,NSTN
      CALL TOTAL(UU(I),VV(I),TE(I),RH(I),ZELE(I),PR(I),MSNG,TEMP(I))
2171  CONTINUE
      YPLT(IT)=AVG(NSTN,TEMP,MSNG)
      GO TO 12
C...FIGURE ABSCISSA.  THE HALF DELTA-T TAKEN CARE OF PREVIOUSLY.
12   T=T+ATH
      XPLT(IT)=T
      WRITE(*,50) IT,TIME,XPLT(IT),YPLT(IT),OPT(ANAL)
50   FORMAT(' I = ',I3,' TIME = ',I4,' X = ',E12.5,' Y = ',
* ,E12.5,1X,A)
C...NEXT TIME.
      CALL NEXT(DATE,TIME,5)
      IF(DATE.GT.EDATE) GO TO 7
      IF(DATE.EQ.EDATE.AND.TIME.GT.ETIME) GO TO 7
      GO TO 8
C...FINALIZE THINGS.

C$$
C. . .ODG ADDITION
7   IF (PLOT) THEN
      XSX=24
      IF(XPLT(IT).GE.24) XSX=48
      YSN = PMIN(ANAL)
      YSX = PMAX(ANAL)
      WRITE(AVGTIME,'(A12,I4,A4)') ':AVG TIME = ',AT,' MIN'
      READ(OPT(ANAL),'(10A4)') (TX(1,J),J=1,10)
      READ(AVGTIME,'(5A4)') (TX(1,J),J=11,15)
      WRITE(NUMSTA,'(I3,A9)') NSTN,' STATIONS'
      WRITE(TSPAN,'(A22,I6,A1,I4,A3,I6,A1,I4,A1)')
-   'START-END DATE/TIME = ',SDATE,'/',STIME,' - ',EDATE,'/',
-   ETIME,' '
      READ(TSPAN,'(12A4)') (TX(2,J),J=1,12)
      READ(NUMSTA,'(3A4)') (TX(2,J),J=13,15)
      DO 21 I = 1, 15
        IF (I .LE. NSTN) THEN
          WRITE(TX(3,I),'(A1,A3)') ' ',STN(I)
        ELSE
          TX(3,I) = ' '
        ENDIF
      ENDIF

```

21 CONTINUE

```
C##### CALL XY3PLT(IT,XPLT,YPLT,MSNG,XSN,XSX,YSN,YSX,LF,LU,LD,LP,TX,IP)
          CALL PMOUT(IT,XPLT,YPLT,MSNG,XSN,XSX,YSN,YSX,LF,LU,LD,LP,TX,IP)
          ENDIF
```

C\$\$

C...A PRINTED DUMP OF RESULTS.

```
WRITE(6,'(//,1X,A/,1X,A,I6,A,I4,A,I6,A,I4/,A,I2,A)') OPT(ANAL)
*, 'START DATE/TIME = ',SDATE,'/',STIME,' END DATE/TIME = ',EDATE
*, '/',ETIME,' AVERAGING TIME = ',AT,' MIN.'
WRITE(6,'(1X,I2,A@)') NSTN,' STATIONS: '
WRITE(6,'(5(A,1X),/,13X,5(A,1X),/,13X,5(A,1X))') (STN(I),I=1,NSTN)
WRITE(6,100) ' _____
```

*

```
DO 22 J=1,IT,7
```

```
I1=J
```

```
I2=J+6
```

```
IF(I2.GT.IT) I2=IT
```

```
WRITE(6,'(2H I,4X,7(I5,6X))') (I,I=I1,I2)
```

```
WRITE(6,'(2H X,7E11.4)') (XPLT(I),I=I1,I2)
```

```
WRITE(6,'(2H Y,7E11.4)') (YPLT(I),I=I1,I2)
```

```
WRITE(6,100) ' _____
```

*

22 CONTINUE

```
GO TO 10
```

220 WRITE(6,102) ' PMKIN terminated. Run PMKINPLT for plots'

```
STOP
```

```
END
```

Option titles

C*****SUBROUTINE TITLES(OPT,NOPT)

C-----HEIMBACH, SEPTEMBER, 1983-----

SUBROUTINE TITLES(OPT,NOPT)

C...SUBROUTINE TO TAKE CLUTTER OUT OF TITLE ORGANIZATION IN PMKIN.

C...ARGUMENTS: OPT = CHAR*40 ARRAY OF OPTIONS.

C NOPT = NUMBER OF OPTIONS AVAILABLE.

```
CHARACTER OPT(1)*40
```

```
NOPT=20
```

```
OPT(1)='STATION PRESSURE (MB)'
```

```
OPT(2)='TEMPERATURE (C)'
```

```
OPT(3)='RELATIVE HUMIDITY'
```

```
OPT(4)='EASTWARD COMP. OF WIND (MPS)'
```

```
OPT(5)='NORTHWARD COMP. OF WIND (MPS)'
```

```
OPT(6)='PRECIPITATION (MM)'
```

```
OPT(7)='SEA LEVEL PRESSURE (MB)'
```

```
OPT(8)='POTENTIAL TEMPERATURE (K)'
```

```
OPT(9)='EQUIV. POT. TEMP. (K)'
```

```
OPT(10)='SPECIFIC HUMIDITY (0/0)'
```

```

OPT(11)='WIND DIRECTION (DEG)'
OPT(12)='TOTAL WIND SPEED (MPS)'
OPT(13)='CONVERGENCE (SEC**-1)'
OPT(14)='RELATIVE VORTICITY (SEC**-1)'
OPT(15)='MOISTURE CONV. (GMS M**-3 SEC**-1)'
OPT(16)='DELTA STATION PRESSURE'
OPT(17)='TOTAL ATM. ENERGY (CAL/GM)'
OPT(18)='START FROM BEGINNING'
OPT(19)='NEW STATIONS ONLY'
OPT(20)='QUIT'
RETURN
END

```

Segmentation communication dummy

```

C*****SUBROUTINE PMOUT(N,X,Y,VM,XSN,XSX,YSN,YSX,LF,LU,LD,LP,TX,IP)
C SEPT 1984 ODG
C ARG LIST IDENTICAL TO XY3PLT. WRITES ARG TO BIN FILE $PMKIN.nnn
C (nnn=COUNT). FILE $PMKIN.000 (ASCII) CONTAINS COUNT ONLY.
      SUBROUTINE PMOUT(N,X,Y,VM,XSN,XSX,YSN,YSX,LF,LU,LD,LP,TX,IP)
      CHARACTER LF*14,TX(3,15)*4,OUTFIL*14,DXCHR*1
      INTEGER    N,LU,LD,LP,IP(10)
      REAL       X(865),Y(865),VM,XSN,XSX,YSN,YSX
      DATA      NOUT/0/
      DXCHR = CHAR(LD+64)
      NOUT = NOUT + 1
      WRITE(OUTFIL,'(A1,IH:,A10)') DXCHR,'$PMKIN.000'
      OPEN(LU,FILE=OUTFIL,STATUS='NEW')
      WRITE(LU,'(I3)') NOUT
      CLOSE(LU)
      WRITE(OUTFIL,'(A1,IH:,A7,I3.3)') DXCHR,'$PMKIN.',NOUT
      OPEN(LU,FILE=OUTFIL,STATUS='NEW',FORM='BINARY')
      WRITE(LU) N
      WRITE(LU) (X(J),Y(J),J=1,N)
      WRITE(LU) VM,XSN,XSX,YSN,YSX
      WRITE(LU) LF,LU,LD,LP,TX,IP
      CLOSE(LU)
      RETURN
      END

```

Date/time and time increment

```

$NOFLOATCALLS
C-----PLIB COLLECTION TO SUPPORT PMKIN-----
C*****SUBROUTINE NEXT(DATE,TIME,DT)
C-----HEIMBACH, SEPTEMBER,1983-----[FEBRUARY,1986]
      SUBROUTINE NEXT(DATE,TIME,DT)
C...SUBROUTINE TO FIND DATE/TIME + TIME INCREMENT.

```

```

C...LEAP YEAR AND CHANGING YEAR NOT FOOLED WITH IN THIS QUICK AND DIRTY JOB.
C...ARGUMENTS: DATE = DATE, YYMMDD,
C              TIME = TIME, HHMM, AND
C              DT = TIME INCREMENT IN MINUTES.
      INTEGER DATE, TIME, DT
      INTEGER*2 YEAR, MONTH, DAY, HOUR, MIN, DAYS(12)
      DATA DAYS(1), DAYS(2), DAYS(3), DAYS(4), DAYS(5), DAYS(6), DAYS(7)
      *, DAYS(8), DAYS(9), DAYS(10), DAYS(11), DAYS(12)/31, 28, 31, 30, 31, 30, 31
      *, 31, 30, 31, 30, 31/
      YEAR=DATE/10000
      MONTH=(DATE/100)-(100.*YEAR)
      DAY=DATE-(10000.*YEAR)-(MONTH*100)
      HOUR=TIME/100
      MIN=TIME-HOUR*100
      MIN=MIN+DT
C...HOOR CHANGE.
      IF(MIN.GT.59) THEN
          MIN=MIN-60
          HOUR=HOUR+1
C...DATE CHANGE.
      IF(HOUR.GT.23) THEN
          HOUR=HOUR-24
          DAY=DAY+1
C...MONTH CHANGE.
      IF(DAY.GT.DAYS(MONTH)) THEN
          DAY=1
          MONTH=MONTH+1
C...INSERT LEAP YEAR AND CHANGE OF YEAR INTO THIS POINT IF SO MOVED.
      ENDIF
      ENDIF
      ENDIF
      DATE=(YEAR*10000)+(MONTH*100)+DAY
      TIME=(HOUR*100)+MIN
      RETURN
      END

```

Kinematic convergence

```

C*****SUBROUTINE PCON(N,XP,YP,UU,VV,MSNG,CON)
C-----HEIMBACH, SEPTEMBER, 1983-----
      SUBROUTINE PCON(N,XP,YP,UU,VV,MSNG,CON)
C...SUBROUTINE TO FIND CONVERGENCE = (1/A)(DA/DT) FROM N STATIONS ENCOMPASSING
C...THE AREA.
C...SEE POOR MAN'S KENEMATICS FOLDER.
C...ARGUMENTS: N = NUMBER OF POINTS,
C              XP, YP = CARTESIAN COORDINATES OF POINTS,
C              UU, VV = EASTWARD AND WESTWARD COORDINATES OF POINTS (MPS),
C              MSNG = MISSING CODE, AND
C              CON = RETURNED CONVERGENCE (SEC**-1).

```

```

        DIMENSION XR(51),YR(51),XP(1),YP(1),UU(1),VV(1),XN(51),YN(51)
        REAL MSNG
C...CHECK FOR MISSING DATA FIRST.
        DO 2 I=1,N
        IF(UU(I).EQ.MSNG.OR.VV(I).EQ.MSNG) THEN
            CON=MSNG
            RETURN
        ENDIF
2      CONTINUE
C...GET ORIGONAL AREA.
        CALL PAREA(N,XP,YP,XR,YR,A)
C      WRITE(*,'(1X,E10.3)') A
C...FIGURE AREA CHANGE OVER ONE HOUR PERIOD.
        DO 1 I=1,N
        XN(I)=XP(I)+UU(I)*3600.
        YN(I)=YP(I)+VV(I)*3600.
C      WRITE(*,'(15,4E10.3)') I,XN(I),YN(I),UU(I),VV(I)
1      CONTINUE
        CALL PAREA(N,XN,YN,XR,YR,AP)
        CON=-(AP-A)/(3600.*A)
C      WRITE(*,'(1X,2E10.3)') AP,CON
        RETURN
    END

```

Area determination

```

C*****SUBROUTINE PAREA(N,XP,YP,XR,YR,AREA)
C-----HEIMBACH, SEPTEMBER, 1983-----
        SUBROUTINE PAREA(N,XP,YP,XR,YR,AREA)
C...SUBROUTINE TO FIND AREA BOUNDED BY N POINTS AS WELL AS ORDER THE POINTS
C...    IN COUNTER-CLOCKWISE FASHION.
C...SEE NOTES DATED 830809; POOR MAN'S KENEMATICS.
C...ARGUMENTS: N = NUMBER OF POINTS,
C              XP, YP = CARTESIAN COORDINATES OF POINTS (RETURNED AREA HAS
C                      SAME UNITS,
C              XR, YR = POINTS ORDERED BY DIRECTION COUNTER-CLOCKWISE FROM
C                      NORTH USING CENTER AS REFERENCE, AND
C              AREA = RETURNED AREA.
        DIMENSION XP(1),YP(1),XR(1),YR(1),TH(50)
C...FIRST FIND AVERAGE X, Y FOR USE AS CENTRAL POINT.
        XB=0
        YB=0
        DO 1 I=1,N
        XB=XB+XP(I)
        YB=YB+YP(I)
1      CONTINUE
        XB=XB/N
        YB=YB/N
C...FIND RADIALS FROM CENTER TO POINTS.  TEMP STORAGE OF ANGLES IN TH.

```

```

DO 2 I=1,N
DX=XP(I)-XB
DY=YP(I)-YB
T=ATAN2(DX,DY)
IF(T.LT.0) T=T+6.28319
TH(I)=T
2   CONTINUE
C...ORDER POINTS WITH DECREASING ANGLE FROM NORTH.
DO 3 I=1,N
T=-1000
DO 4 J=1,N
IF(T.GT.TH(J)) GO TO 4
K=J
T=TH(J)
4   CONTINUE
TH(K)=-2000
XR(I)=XP(K)
YR(I)=YP(K)
3   CONTINUE
XR(N+1)=XR(1)
YR(N+1)=YR(1)
C...WORK UP AREAS.
AREA=0
DO 5 I=1,N
A2=((XR(I)-XB)**2)+((YR(I)-YB)**2)
B2=((XR(I+1)-XB)**2)+((YR(I+1)-YB)**2)
C2=((XR(I+1)-XR(I))**2)+((YR(I+1)-YR(I))**2)
C=SQRT(C2)
DAREA=(A2-(((A2+C2-B2)/(2.*C))**2))
IF(DAREA.LT.0) THEN
    WRITE(*,6) ' PAREA WARNING -- NEGATIVE SQUARE ROOT OF '
*   ,DAREA,'      A2 = ',A2,'      B2 = ',B2,'      C2 = ',C2
*   , ' DAREA SET TO ZERO.'
6   FORMAT(A,E12.5,/,A,E12.5,A,E12.5,A,E12.5,/,A)
    DAREA=0.0
    GO TO 5
ENDIF
DAREA=ABS(C)*SQRT(DAREA)/2.0
AREA=AREA+DAREA
5   CONTINUE
RETURN
END

```

Pressure change

```

C*****SUBROUTINE DELTAP(DELP,NSTN,MSNG,DP)
C-----HEIMBACH, NOVEMBER, 1983-----
      SUBROUTINE DELTAP(DELP,NSTN,MSNG,DP)
C...SUBROUTINE TO PROVIDE DELTA-P'S USING FIRST AND LAST PRESSURES IN SPAN.

```

```

C...ARGUMENTS: DELP(I,J) = PRESSURE FOR STATION I,
C                J = 1 FOR FIRST AND 2 FOR LAST.
C                NSTN = NUMBER OF STATIONS.
C                MSNG = MISSING CODE.
C                DP = AVERAGE TREND RETURNED.
      REAL MSNG,DELP(50,2)
      DP=0.0
      DO 1 I=1,NSTN
      IF(DELP(I,1).EQ.MSNG.OR.DELP(I,2).EQ.MSNG) THEN
        DP=MSNG
        RETURN
      ENDIF
      DP=DP+(DELP(I,2)-DELP(I,1))
1     CONTINUE
      DP=DP/NSTN
      RETURN
      END

```

Lambert conformal projection distances

```

C*****SUBROUTINE LAMB(LATR, LONGR, LATF, LONGF, LAT, LONG, XF, YF, X, Y, FIRST)
C-----HEIMBACH, SEPTEMBER, 1983-----
      SUBROUTINE LAMB(LATR, LONGR, LATF, LONGF, LAT, LONG, XF, YF, X, Y, FIRST)
C...SUBROUTINE TO FIND CARTESIAN DISTANCES FROM A FIDUCIAL POINT USING THE
C...LAMBERT CONFORMAL MAP PROJECTION. SEE THEORETICAL NOTES, P 32 SAUCIER
C...OR PMKIN NOTES DATED 830902.
C...X IS TO THE EAST OF THE FIDUCIAL POINT IN METERS, Y IS TO NORTH.
C
C...ARGUMENTS: LATR, LONGR = REFERENCE LAT AND LONG (DEG),
C                LATF, LONGF = LAT AND LONG OF FIDUCIAL POINT (SW CORNER OF MAP,
C                LAT, LONG = LAT AND LONG OF DATA POINT.
C                XF, YF = CARTESIAN DISTANCES OF FIDUCIAL POINT FROM PROJECTED
C                NORTH POLE,
C                X, Y = CARTESIAN DISTANCES OF DATA POINT FROM FIDUCIAL POINT, AND
C                FIRST = .TRUE., .FALSE. IMPLIES FIGURE FIDUCIAL POINT, FIGURE
C                X, Y (LOGICAL).
      IMPLICIT REAL (I)
      DOUBLE PRECISION A,B,C,D,E,F,R,DXF,DYF,RAD,RAD2,N,H
      LOGICAL FIRST
      DATA RAD/0.017453293/,RAD2/0.008726646/
      IF(FIRST) THEN
        A=(90.-LATR)*RAD
        B=6.370E+06*DTAN(A)
        C=DTAN(A/2.)
        D=DCOS(A)
        N=DSIN(LATR*RAD)
        F=(LONGF-LONGR)*RAD*N
        H=(90.-LATF)*RAD2
        H=B*((DTAN(H)/C)**D)

```

```

DXF=H*DSIN(F)
DYF=H*DCOS(F)
XF=DXF
YF=DYF
ELSE
E=(LONG-LONGR)*RAD*N
R=(90.-LAT)*RAD2
R=B*((DTAN(R)/C)**D)
X=DXF-R*DSIN(E)
Y=DYF-R*DCOS(E)
ENDIF
RETURN
END

```

Vorticity

```

C*****SUBROUTINE PVORT(N,XP,YP,UU,VV,MSNG,VORT)
C-----HEIMBACH, SEPTEMBER, 1983-----
      SUBROUTINE PVORT(N,XP,YP,UU,VV,MSNG,VORT)
CALLS PCON
C...SUBROUTINE TO FIND VORTICITY FROM N STATIONS ENCOMPASSING AN AREA.
C...SEE POOR MAN'S KINEMATIC FOLDER OR SAUCIER P. 339.
C...ARGUMENTS: N = NUMBER OF POINTS.
C              XP, YP = CARTESIAN COORDINATES OF POINTS,
C              UU, VV = EASTWARD AND WESTWARD WIND SPEEDS (MPS),
C              MSNG = MISSING CODE, AND
C              VORT = CALCULATED VORTICITY (SEC**-1).
C
      DIMENSION XP(1),YP(1),UU(1),VV(1)
      REAL MSNG
C...ROTATE WINDS 90 DEGREES TO FIND VORTICITY. AT SAME TIME CHECK FOR
C...MISSING DATA.
      DO I I=1,N
      IF(UU(I).EQ.MSNG.OR.VV(I).EQ.MSNG) THEN
        VORT=MSNG
        RETURN
      ENDIF
      T=-UU(I)
      UU(I)=VV(I)
      VV(I)=T
1    CONTINUE
      CALL PCON(N,XP,YP,UU,VV,MSNG,VORT)
C...PCON RETURNS CONV=-DIV.
      VORT=-VORT
      RETURN
END

```

Convert wind components into direction and speed

```

C*****SUBROUTINE WIND(UU,VV,MSNG,DD,FF)
C-----HEIMBACH, SEPTEMBER, 1983-----
      SUBROUTINE WIND(UU,VV,MSNG,DD,FF)
C...SUBROUTINE TO CONVERT WIND COMPONENTS INTO DIRECTION AND SPEED.
C...ARGUMENTS: UU = EASTWARD COMPONENT,
C              VV = NORTHWARD COMPONENT,
C              MSNG = MISSING CODE,
C              DD = RETURNED DIRECTION, AND
C              FF = RETURNED SPEED
      REAL MSNG
      IF(UU.EQ.MSNG.OR.VV.EQ.MSNG) THEN
        DD=MSNG
        FF=MSNG
        RETURN
      ENDIF
      IF(UU.EQ.0.0) UU=1.0E-15
      DD=ATAN2(VV,UU)
      DD=270.-DD*57.29578
      IF(DD.GT.360.) DD=DD-360.
      FF=SQRT((UU*UU)+(VV*VV))
      RETURN
      END

```

Average of N parameters

```

C*****FUNCTION AVG(N,PAR,MSNG)
C-----HEIMBACH, SEPTEMBER, 1983-----
      FUNCTION AVG(N,PAR,MSNG)
C...FUNCTION TO FIND SIMPLE AVERAGE OF N PARAMETERS.
C...MSNG = MISSING CODE. REQUIRE ALL POINTS TO BE PRESENT.
      REAL MSNG,PAR(1)
      AV=0.0
      DO 1 I=1,N
        IF(PAR(I).EQ.MSNG) THEN
          AVG=MSNG
          RETURN
        ELSE
          AV=AV+PAR(I)
        ENDIF
      CONTINUE
      AVG=AV/N
      RETURN
      END

```

Specific humidity and mixing ratio

```

C*****SUBROUTINE HUM(TE,RH,PR,MSNG,SH,AH)
C-----HEIMBACH, SEPTEMBER, 1983-----
      SUBROUTINE HUM(TE,RH,PR,MSNG,SH,AH)
C...FUNCTION TO FIND MIXING RATIO AND SPECIFIC HUMIDITY
C...HUMIDITY.  SEE SAUCIER PP. 9-11, 830923 NOTES.
C...ARGUMENTS: TE = TEMPERATURE (C),
C               RH = RELATIVE HUMIDITY (ENG. UNITS.),
C               PR = PRESSURE (MB),
C               MSNG = MISSING CODE,
C               SH = SPECIFIC HUMIDITY (0/0), AND
C               AH = ABSOLUTE HUMIDITY (GM M**-3).
C...TABATA'S SECOND ORDER EQUATION USED FOR SAT VAP PRES.
      REAL MR,MSNG
      IF(TE.EQ.MSNG.OR.RH.EQ.MSNG.OR.PR.EQ.MSNG) THEN
        MR=MSNG
        SH=MSNG
        RETURN
      ENDIF
      T=1000.0/(TE+273.16)
      E=8.4293-(1.8272*T)-(0.071208*T*T)
      E=10.0**E
      E=E*RH
      SH=0.62197*E/(PR-(0.37803*E))
      AH=216.54*E/(TE+273.16)
C  WRITE(*,'(1X,6E10.3)') TE,RH,PR,E,SH,AH
      RETURN
      END

```

Potential and equivalent potential temperature

```

C*****SUBROUTINE POT(TE,RH,PR,MSNG,TH,THE)
CALLS HUM
C-----HEIMBACH, SEPTEMBER, 1983-----
      SUBROUTINE POT(TE,RH,PR,MSNG,TH,THE)
C...SUBROUTINE TO FIND POTENTIAL AND EQUIVALENT POTENTIAL TEMPERATURES.
C...SEE PP. 9 - 14 SAUCIER.
C...ARGUMENTS: TE = TEMPERATURE (C),
C               RH = RELATIVE HUMIDITY (ENG. UNITS),
C               PR = PRESSURE IN MB,
C               MSNG = MISSING CODE (REAL),
C               TH = POTENTIAL TEMP (K), AND
C               THE = EQUIVALENT POTENTIAL TEMPERATURE (K).
      REAL MSNG
C...SUBROUTINE HUM LOOKS FOR MSNG.
      TH=MSNG

```

```

THE=MSNG
CALL HUM(TE,RH,PR,MSNG,Q,AH)
IF(TE.NE.MSNG.AND.PR.NE.MSNG) THEN
    TK=TE+273.16
    TH=TK*((1000./PR)**0.285714)
ELSE
    RETURN
ENDIF
C...ASSUME MIXING RATIO APPROX. EQUAL TO SPECIFIC HUMIDITY.
IF(Q.NE.MSNG) THE=TH*EXP(2250.*Q/TK)
RETURN
END

```

Sea level pressure

```

C*****FUNCTION PSEA(ZS,TS,PS,MSNG)
C-----HEIMBACH, SEPTEMBER, 1983-----
    FUNCTION PSEA(ZS,TS,PS,MSNG)
C...FUNCTION TO FIND SEA LEVEL PRESSURE. SEE PP. 56 - 58 SAUCIER.
C...ARGUMENTS: ZS = SURFACE ELEVATION (M),
C               TS = SURFACE TEMPERATURE,
C               PS = SURFACE PRESSURE (MB), AND
C               MSNG = MISSING CODE.
    REAL MSNG
    IF(ZS.EQ.MSNG.OR.TS.EQ.MSNG.OR.PS.EQ.MSNG) THEN
        PSEA=MSNG
        RETURN
    ENDIF
    PSEA=PS*EXP(ZS/(29.28980*((TS+273.16)+(ZS/400))))
    RETURN
END

```

Moisture convergence

```

C*****SUBROUTINE PMCON(N,XP,YP,PR,TE,RH,UU,VV,MSNG,MCON)
CALLS PCON
CALLS HUM
    SUBROUTINE PMCON(N,XP,YP,PR,TE,RH,UU,VV,MSNG,MCON)
C...SUBROUTINE TO ESTIMATE, AND I MEAN CRUDELY, MOISTURE CONVERGENCE FROM
C...SEVERAL STATIONS. SEE NOTES DATED 830923. ALSO INTO TO DYN MET,
C...HESS, LEICHTER & DENNIS '74, ETC. ETC.
C...ARGUMENTS: N = NUMBER OF STATIONS,
C               XP, YP = CARTESIAN COORDINATES OF POINTS ON MAP PROJECTION (M),
C               PR = STATION PRESSURE (MB),
C               TE = STATION TEMPERATURE (C),
C               RH = STATION HUMIDITIES (ENGR),
C               UU, VV = EASTWARD AND NORTHWARD WIND SPEEDS (MPS),
C               MSNG = MISSING CODE, AND

```

```

C          MCON = RETURNED MOISTURE CONVERGENCE (GM, M**3, SEC**-1).
          DIMENSION XP(1),YP(1),PR(1),TE(1),RH(1),UU(1),VV(1)
          REAL MSNG,MCON
C...FIRST CHECK FOR MISSING DATA.
C...MISSING UU AND VV CHECKED FOR IN PCON.
          DO 1 I=1,N
          IF(PR(I).EQ.MSNG.OR.TE(I).EQ.MSNG.OR.RH(I).EQ.MSNG) THEN
              MCON=MSNG
              RETURN
          ENDIF
1          CONTINUE
C...GET ABSOLUTE HUMIDITIES.  ASSUME ALL PARAMETERS ARE PRESENT AFTER
C...ABOVE TEST.
          AVE=0.0
          DO 2 I=1,N
          CALL HUM(TE(I),RH(I),PR(I),MSNG,DUM,AH)
          AVE=AVE+AH
2          CONTINUE
          AVE=AVE/N
C...GET HORIZONTAL VELOCITY CONVERGENCE.
          CALL PCON(N,XP,YP,UU,VV,MSNG,CON)
          IF(CON.EQ.MSNG). THEN
              MCON=MSNG
              RETURN
          ENDIF
          MCON=AVE*CON
C          WRITE(*,'(1X,3E12.4)') AVE,CON,MCON
          RETURN
          END

```

Running means

```

C*****SUBROUTINE RMWRK(NSTN,NRM,MSNG)
C...SUBROUTINE TO WORK UP RUNNING MEANS AND DO SOME BOOK KEEPING.  IDEA
C...IS TO KEEP PMKIN MAINLINE UNCLUTTERED BY SUCH BUSY WORK.
C...A MEAN OF 5 PARAMETERS IS FOUND FOR STATION NSTN.  AFTER FINDING THE
C...MEAN, THAT SET IS LAGGED BY ONE TO MAKE ROOM FOR THE NEXT INPUT FOR THAT
C...STATION.
          SUBROUTINE RMWRK(NSTN,NRM,MSNG)
C...ARGUMENTS RM(L,M,N) = STASH ARRAY (SEE Labeled COMMON),
C          L = STATION INDEX,
C          M = PARAMETER INDEX,
C          N = TIME INDEX,
C          NSTN = STATION NUMBER
C          NRM = NUMBER OF PERIODS IN RUNNING AVERAGE,
C          MSNG = MISSING CODE, AND
C          AV = 5 RETURNED AVERAGES (SEE Labeled COMMON).
          REAL MSNG
          COMMON /RMAVG/ RM(50,5,12),AV(5)

```

```

DO 1 M=1,5
AV(M)=0
DO 2 N=1,NRM
IF(RM(NSTN,M,N).EQ.MSNG) THEN
    AV(M)=MSNG
    GO TO 1
ENDIF
AV(M)=AV(M)+RM(NSTN,M,N)
2 CONTINUE
AV(M)=AV(M)/NRM
1 CONTINUE
C...SHIFT BACK ONE DELTA-T.
IF(NRM.EQ.1) RETURN
DO 3 M=1,5
DO 3 N=2,NRM
RM(NSTN,M,N-1)=RM(NSTN,M,N)
3 CONTINUE
RETURN
END

```

Total specific atmospheric energy

```

C*****SUBROUTINE TOTAL(UU,VV,T,RH,Z,P,MSNG,TOT)
C...SUBROUTINE TO FIND TOTAL ATMOSPHERIC ENERGY USING THEORY BY DARKOW
SUBROUTINE TOTAL(UU,VV,T,RH,Z,P,MSNG,TOT)
CALLS HUM
C...ARGUMENTS: UU, VV = EASTWARD AND NORTHWARD WIND CONPONNENTS (M/S),
C                T = TEMPERATURE (C),
C                RH = RELATIVE HUMIDITY (0/0),
C                Z = STATION ELEVATION (M),
C                P = STATION PRESSURE (MB),
C                MSNG = MISSING CODE (REAL), AND
C                TOT = TOTAL ENERGY (CAL/GM).
C
REAL MSNG
C...FIRST CHECK FOR MISSING DATA. SUBROUTINE HUM DHECKS T, P, AND RH.
TOT=MSNG
IF(Z.EQ.MSNG.OR.UU.EQ.MSNG.OR.VV.EQ.MSNG) RETURN
CALL HUM(T,RH,P,MSNG,Q,AH)
IF(Q.EQ.MSNG) RETURN
TOT=(UU*UU+VV*VV)/8372+(T+273.16)*(0.240+(0.201*Q))
*      +(667*Q)+(0.00234*Z)
C...TEST DUMP
C WRITE(*,'(1X,7E11.4)') UU,VV,T,RH,Z,P,MSNG,TOT,Q,AH
RETURN
END

```

LSA Intermediate SegmentPMKIN plot preprocessor

```

C*****PROGRAM PMKPP
C JULY 1985 ODGreen -- PMKPP : PREPARE PMKIN PLOT FILES FOR XY3PLT
C THE XY3PLT SCALING FUNCTIONS ARE DEPENDENT ON SEVERAL USER-
C DEFINED PARAMETERS AS WELL AS THE ACTUAL DATA. THUS, IT IS
C NOT ALWAYS READILY EVIDENT WHAT LIMITS AND STEP SIZE WILL
C APPEAR ON THE ORDINATE SCALE. PMKPP ALLOWS THE USER TO
C EXAMINE THE RESULTS OF VARIOUS INPUTS USING EITHER REAL (EG,
C $PMKIN.NNN TYPE FILES) OR HYPOTHETICAL (USER-ENTERED) DATA.
C FEBRUARY 1986 ODGreen -- CLIP OPTIONS ADDED
C
C LINK STREAM :
C
C PROGRAM PMKPP
C
C***DECLARATIONS
C CHARACTER LF*14, TX(3,15)*4, INFIL*14, OUTFIL*14, ANS*1
C INTEGER N, LU, LD, LP, IP(10), IPD(10)
C REAL X(865), Y(865), VM, XSN, XSX, YSN, YSX, LIM(4), UX(2), UY(2)
C
C EQUIVALENCE (LIM(1), XSN), (LIM(2), XSX), (LIM(3), YSN), (LIM(4), YSX)
C DATA IPD /2,8,24,2,3,8,10,1,8,4/
C
C***FORMAT STATEMENTS
C 90000 FORMAT(A)
C 90010 FORMAT(A,*)
C 90020 FORMAT(I2)
C 90030 FORMAT(A14)
C 90040 FORMAT(1X, 'FILE: ', A14, 1X, A6)
C 90050 FORMAT(1X, A1, '(', I1, ') = ? ', *)
C 90060 FORMAT(E13.4)
C 90070 FORMAT(A, I2, A, *)
C 90080 FORMAT(A, I2)
C 90090 FORMAT(A, E13.4)
C
C***BEGIN PMKPP
C WRITE(*, 90000) 'OPMKPP -- PREPARE PMKIN PLOT FILES FOR XY3PLT'
C
C***SELECT REAL/HYPOTHETICAL DATA MODES (OR EXIT PROGRAM)
C 10000 WRITE(*, 90000) 'OSELECT DATA MODE:'
C WRITE(*, 90000) ' 1) USE REAL DATA'
C WRITE(*, 90000) ' 2) USE HYPOTHETICAL DATA'
C WRITE(*, 90010) ' ENTER ITEM NUMBER OR ZERO :'
C READ (*, 90020) IANS
C IF ( (IANS .LT. 0) .OR. (IANS .GT. 2) ) GOTO 10000
C GOTO (99999, 20000, 30000) IANS + 1

```

C

C***INPUT SOURCE/DESTINATION FILESPECS

```

20000 WRITE(*,90010) 'OENTER SOURCE FILESPEC OR ¼CR½ : '
      READ (*,90030) INFIL
      IF (INFIL .EQ. ' ') GOTO 10000
      WRITE(*,90010) ' ENTER DESTINATION FILESPEC      : '
      READ (*,90030) OUTFIL
      IF (OUTFIL .EQ. ' ') GOTO 10000

```

C

C***READ THE SOURCE FILE DATA

```

21000 OPEN(1,FILE=INFIL,STATUS='OLD',FORM='BINARY')
      WRITE(*,90040) INFIL,'OPENED'
      READ(1) N
      READ(1) (X(J),Y(J),J=1,N)
      READ(1) VM,XSN,XSX,YSN,YSX
      READ(1) LF,LU,LD,LP,TX,IP
      CLOSE(1)
      WRITE(*,90040) INFIL,'CLOSED'
      DO 21050 I = 1, 10
        IF (IP(I) .LE. 0) IP(I) = IPD(I)
21050 CONTINUE
      GOTO 50000

```

C

C***INPUT HYPOTHETICAL DATA

```

30000 WRITE(*,90000) 'OENTER HYPOTHETICAL DATA (2 X,Y PAIRS): '
      DO 30010 I = 1, 2
        WRITE(*,90050) 'X',I
        READ (*,90060) UX(I)
        WRITE(*,90050) 'Y',I
        READ (*,90060) UY(I)
30010 CONTINUE
      WRITE(*,90010) 'OMISSING VALUE ? '
      READ (*,90060) VM
      DO 30020 I = 1, 10
        IP(I) = IPD(I)
30020 CONTINUE

```

C

C***INPUT SCALE LIMITS

```

31000 WRITE(*,90000) 'OENTER PLOT LIMITS : '
      WRITE(*,90010) ' MIN X PLOT VALUE ? '
      READ (*,90060) XSN
      WRITE(*,90010) ' MAX X PLOT VALUE ? '
      READ (*,90060) XSX
      WRITE(*,90010) ' MIN Y PLOT VALUE ? '
      READ (*,90060) YSN
      WRITE(*,90010) ' MAX Y PLOT VALUE ? '
      READ (*,90060) YSX

```

C

C***EDIT, VIEW, WRITE OUT TO FILE ?

```

40000 WRITE(*,90000) 'OSELECT OPTION: '

```

```

WRITE(*,90000) ' 1) EDIT MODE'
WRITE(*,90000) ' 2) VIEW RESULTS'
IF (IANS .EQ. 1) WRITE (*,90000) ' 3) WRITE OUT TO FILE'
WRITE(*,90010) ' ENTER ITEM NUMBER OR ZERO :'
READ (*,90020) JANS
IF ((IANS.NE.1).AND.((JANS.LT.0).OR.(JANS.GT.2))) GOTO 40000
IF ((IANS.EQ.1).AND.((JANS.LT.0).OR.(JANS.GT.3))) GOTO 40000
IF (IANS .NE. 1) GOTO (10000,50000,60000) JANS + 1
IF (IANS .EQ. 1) GOTO (10000,50000,60000,70000) JANS + 1

```

C

C***GET EDIT ITEM

```

50000 WRITE(*,90000) '0SELECT EDIT ITEM:'
WRITE(*,90000) ' 1) LOGFILE DRIVE'
WRITE(*,90000) ' 2) IP ARRAY'
WRITE(*,90000) ' 3) PLOT LIMITS'
WRITE(*,90000) ' 4) CLIP OPTIONS'
WRITE(*,90010) ' ENTER ITEM NUMBER OR ZERO:'
READ (*,90020) JANS
IF ( (JANS .LT. 0) .OR. (JANS .GT. 4) ) GOTO 50000
GOTO (40000,51000,52000,53000,54000) JANS + 1

```

C

C***INPUT LOGFILE DRIVE

```

51000 WRITE(*,90070) 'OLD=',LD,' LD=?'
READ (*,90020) LD
GOTO 50000

```

C

C***EDIT IP ARRAY

```

52000 WRITE(*,90000) '0SELECT IP ITEM TO EDIT:'
WRITE(*,90080) ' 1) #DIG X SCALE LABEL = ',IP(1)
WRITE(*,90080) ' 2) SIZE X SCALE LABEL = ',IP(2)
WRITE(*,90080) ' 3) #TICK MARKS X SCALE = ',IP(3)
WRITE(*,90080) ' 4) FREQUENCY X LABEL = ',IP(4)
WRITE(*,90080) ' 5) #DIG Y SCALE LABEL = ',IP(5)
WRITE(*,90080) ' 6) SIZE Y SCALE LABEL = ',IP(6)
WRITE(*,90080) ' 7) #TICK MARKS Y SCALE = ',IP(7)
WRITE(*,90080) ' 8) FREQUENCY Y LABEL = ',IP(8)
WRITE(*,90080) ' 9) SIZE OF TITLE LABEL = ',IP(9)
WRITE(*,90080) ' 10) SIZE OF TICK MARK = ',IP(10)
WRITE(*,90010) ' ENTER ITEM NUMBER OR ZERO :'
READ (*,90020) JANS
IF ( (JANS .LT. 0) .OR. (JANS .GT. 10) ) GOTO 52000
IF (JANS .EQ. 0) GOTO 50000
WRITE(*,90010) '0ENTER NEW VALUE :'
READ (*,90020) IP(JANS)
GOTO 52000

```

C

C***EDIT PLOT LIMITS

```

53000 CALL RMXMN1(X,N,VM,XMIN,XMAX)
CALL RMXMN1(Y,N,VM,YMIN,YMAX)
WRITE(*,90000) '0CURRENT DATA EXTREMES :'

```

```

WRITE(*,90090) ' MIN X DATA VALUE = ',XMIN
WRITE(*,90090) ' MAX X DATA VALUE = ',XMAX
WRITE(*,90090) ' MIN Y DATA VALUE = ',YMIN
WRITE(*,90090) ' MAX Y DATA VALUE = ',YMAX

```

C

```

WRITE(*,90000) 'OSELECT PLOT LIMIT TO EDIT : '
WRITE(*,90090) ' 1) MIN X = ',XSN
WRITE(*,90090) ' 2) MAX X = ',XSX
WRITE(*,90090) ' 3) MIN Y = ',YSN
WRITE(*,90090) ' 4) MAX Y = ',YSX
WRITE(*,90010) ' ENTER ITEM NUMBER OR ZERO : '
READ (*,90020) JANS
IF ( (JANS .LT. 0) .OR. (JANS .GT. 4) ) GOTO 53000
IF (JANS .EQ. 0) GOTO 50000
WRITE(*,90010) 'OINPUT NEW LIMIT : '
READ (*,90060) LIM(JANS)
GOTO 53000

```

C

C***CLIP OPTIONS

```

54000 WRITE(*,90000) 'OSELECT CLIP OPTION: '
WRITE(*,90000) ' 1) SPECIFY Y LIMITS FOR ORDINATE CLIP'
WRITE(*,90000) ' 2) SPECIFY X LIMITS FOR ABSCISSA CLIP'
WRITE(*,90000) ' 3) SPECIFY X LIMITS FOR ORDINATE CLIP'
WRITE(*,90000) ' 4) SPECIFY Y LIMITS FOR ABSCISSA CLIP'
WRITE(*,90010) ' ENTER ITEM NUMBER OR ZERO : '
READ (*,90020) JANS
IF ( (JANS .LT. 0) .OR. (JANS .GT. 10) ) GOTO 54000
GOTO (50000,54100,54200,54300,54400) JANS + 1

```

C

C---SPECIFY Y LIMITS FOR ORDINATE CLIP

```

54100 WRITE(*,90000) 'OSPECIFY Y LIMITS FOR ORDINATE CLIP'
WRITE(*,90010) ' INPUT LOWER Y LIMIT : '
READ (*,90060) YLO
WRITE(*,90010) ' INPUT UPPER Y LIMIT : '
READ (*,90060) YHI
WRITE(*,90010) ' =1 CLIP INSIDE LIMITS 1/21 CLIP. OUTSIDE LIMITS?'
READ (*,90020) IOPT
DO 54150 I = 1, N
  IF (IOPT .EQ. 1) THEN
    IF ( (Y(I) .GE. YLO) .AND. (Y(I) .LE. YHI) ) Y(I) = VM
  ELSE
    IF ( (Y(I) .LT. YLO) .OR. (Y(I) .GT. YHI) ) Y(I) = VM
  ENDIF
54150 CONTINUE
GOTO 54000

```

C

C---SPECIFY X LIMITS FOR ABSCISSA CLIP

```

54200 WRITE(*,90000) 'OSPECIFY X LIMITS FOR ABSCISSA CLIP'
WRITE(*,90010) ' INPUT LOWER X LIMIT : '
READ (*,90060) XLO

```

```

WRITE(*,90010) ' INPUT UPPER X LIMIT : '
READ (*,90060) XHI
WRITE(*,90010) ' =1 CLIP INSIDE LIMITS  1/21 CLIP OUTSIDE LIMITS?'
READ (*,90020) IOPT
DO 54250 I = 1, N
  IF (IOPT .EQ. 1) THEN
    IF ( (X(I) .GE. XLO) .AND. (X(I) .LE. XHI) ) X(I) = VM
  ELSE
    IF ( (X(I) .LT. XLO) .OR. (X(I) .GT. XHI) ) X(I) = VM
  ENDIF
54250 CONTINUE
GOTO 54000
C
C---SPECIFY X LIMITS FOR ORDINATE CLIP
54300 WRITE(*,90000) 'OSPECIFY X LIMITS FOR ORDINATE CLIP'
WRITE(*,90010) ' INPUT LOWER X LIMIT : '
READ (*,90060) XLO
WRITE(*,90010) ' INPUT UPPER X LIMIT : '
READ (*,90060) XHI
WRITE(*,90010) ' =1 CLIP INSIDE LIMITS  1/21 CLIP OUTSIDE LIMITS?'
READ (*,90020) IOPT
DO 54350 I = 1, N
  IF (IOPT .EQ. 1) THEN
    IF ( (X(I) .GE. XLO) .AND. (X(I) .LE. XHI) ) Y(I) = VM
  ELSE
    IF ( (X(I) .LT. XLO) .OR. (X(I) .GT. XHI) ) Y(I) = VM
  ENDIF
54350 CONTINUE
GOTO 54000
C
C---SPECIFY Y LIMITS FOR ABSCISSA CLIP
54400 WRITE(*,90000) 'OSPECIFY Y LIMITS FOR ABSCISSA CLIP'
WRITE(*,90010) ' INPUT LOWER Y LIMIT : '
READ (*,90060) YLO
WRITE(*,90010) ' INPUT UPPER Y LIMIT : '
READ (*,90060) YHI
WRITE(*,90010) ' =1 CLIP INSIDE LIMITS  1/21 CLIP OUTSIDE LIMITS?'
READ (*,90020) IOPT
DO 54450 I = 1, N
  IF (IOPT .EQ. 1) THEN
    IF ( (Y(I) .GE. YLO) .AND. (Y(I) .LE. YHI) ) X(I) = VM
  ELSE
    IF ( (Y(I) .LT. YLO) .OR. (Y(I) .GT. YHI) ) X(I) = VM
  ENDIF
54450 CONTINUE
GOTO 54000
C
C***VIEW RESULTS
60000 XNP = XSN
      XXP = XSX

```

```

      YNP = YSN
      YXP = YSX
      IF (IANS .EQ. 1) THEN
        CALL SCAXLR(N,X,VM,XNP,XXP,IP(3),IP(1),XS,IS,ISF)
        CALL SCAXLR(N,Y,VM,YNP,YXP,IP(7),IP(5),YS,JS,JSF)
      ELSEIF (IANS .EQ. 2) THEN
        CALL SCAXLR(3,UX,VM,XNP,XXP,IP(3),IP(1),XS,IS,ISF)
        CALL SCAXLR(3,UY,VM,YNP,YXP,IP(7),IP(5),YS,JS,JSF)
      ENDIF

C
      WRITE(*,90000) 'OABSCISSA SCALE RESULTS:'
      WRITE(*,90080) ' #STEPS           = ',IS
      WRITE(*,90080) ' SCALE FACTOR      = ',ISF
      WRITE(*,90090) ' STEP SIZE         = ',XS
      WRITE(*,90090) ' MIN X              = ',XNP
      WRITE(*,90090) ' MAX X              = ',XXP

C
      WRITE(*,90000) 'OORDINATE SCALE RESULTS:'
      WRITE(*,90080) ' #STEPS           = ',JS
      WRITE(*,90080) ' SCALE FACTOR      = ',JSF
      WRITE(*,90090) ' STEP SIZE         = ',YS
      WRITE(*,90090) ' MIN Y              = ',YNP
      WRITE(*,90090) ' MAX Y              = ',YXP
      GOTO 40000

C
C***WRITE OUT TO FILE
70000 OPEN(1,FILE=OUTFIL,STATUS='NEW',FORM='BINARY')
      WRITE(*,90040) OUTFIL,'OPENED'
      WRITE(1) N
      WRITE(1) (X(J),Y(J),J=1,N)
      WRITE(1) VM,XSN,XSX,YSN,YSX
      WRITE(1) LF,LU,LD,LP,TX,IP
      CLOSE(1)
      WRITE(*,90040) OUTFIL,'CLOSED'
      GOTO 10000

C
C***TERMINATE PMKPP
99999 STOP ' PMKPP -- END OF RUN'
      END

```

Maximum/minimum values

C*****SUBROUTINE RMXMN1 (D,N,DM,DN,DX)

C Version 1.00 (10 October 1983)
C Update

by Orville D. Green

C Description :

C RMXMN returns the minimum and maximum values in a data array, while

C ignoring missing data

C Arguments :

C D [real] : array containing data values
 C N [integer] : number of items in D to compare
 C DM [real] : value assigned to a missing datum
 C DN [real] : minimum D value
 C DX [real] : maximum D value

SUBROUTINE RMXMN1 (D,N,DM,DN,DX)

C declarations

REAL D(1)

C data statements

DATA RX /1.7E38/, RN /-1.7E38/

*Begin RMXMN1

DN = RX

DX = RN

DO 10 I = 1, N

IF (D(I) .NE. DM) DN = AMIN1(D(I),DN)

IF (D(I) .NE. DM) DX = AMAX1(D(I),DX)

10 CONTINUE

IF (DN .EQ. RX) DN = DM

IF (DX .EQ. RN) DX = DM

*Terminate RMXMN1

RETURN

END

LSA Back-endPMKIN plot module

```

C*****PROGRAM PMKINP
C SEPT 1984 ODG -- PMKINPLT: BACKEND FOR PMKIN
C READS $PMKIN.nnn FILES AND SCHEDULES XY3PLT SYSTEM
C COMPILE USING MS-FORTRAN 3.04
C LOGICAL UNITS : 1
C LINK STREAM :
C  GDATE+GTIME+LOOK+PMKINPLT+XY3PLT+&XY3PLT+$XY3PLT,PMKINPLT...
CALLS XY3PLT
C DATAFILES $MSGDATA.BIN, $PMKIN.nnn ON DEFAULT DRIVE
  PROGRAM PMKINP
  CHARACTER LF*14,LF1*14,IX(3,15)*4,INFIL*14,NINFIL*3
  INTEGER N,LU,LD,LP,IP(10),NF,I1(3)
  REAL X(865),Y(865),VM,XSN,XSX,YSN,YSX
  DATA LF/' '/
  WRITE(*,'(A)') 'OPMKINPLT -- PMKIN backend for XY3plots'
  OPEN(1,FILE='$PMKIN.000',STATUS='OLD')
  READ(1,'(I3)') NF
  CLOSE(1)
  DO 10 I = 1, NF
    WRITE(NINFIL,'(I3)') I
    READ(NINFIL,'(3I1)') I1
    WRITE(INFIL,'(A7,3I1)') '$PMKIN.',I1
    OPEN(1,FILE=INFIL,STATUS='OLD',FORM='BINARY')
    READ(1) N
    READ(1) (X(J),Y(J),J=1,N)
    READ(1) VM,XSN,XSX,YSN,YSX
    READ(1) LF1,LU,LD,LP,IX,IP
    CLOSE(1)
    IF(LF1.NE.' ') LF=LF1
    CALL XY3PLT(N,X,Y,VM,XSN,XSX,YSN,YSX,LF,LU,LD,LP,IX,IP)
10  CONTINUE
  STOP 'PMKINPLT termination. Run XY3PRT for hardcopies'
  END

```

APPENDIX C

PLOT PROGRAM LISTINGS

APPENDIX C

PLOT PROGRAM LISTINGS

Plot Hardcopies

```

C=====
C SOURCE FILE: XY3PRT.FOR
C
C 06 JUNE 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
C
C LOCALLY CONTAINED PROGRAM UNITS:
C [P] XY3PRT : OUTPUT XY3PLT CREATED PLOTS
C [S] XY3GPF : GET PLOT FILE INFORMATION FROM LOG FILE
C [S] XY3PPF : SCHEDULE PLOT OUTPUT
C [S] PDNPL : MODIFIED VERSION OF T. ENGEL'S PDNPLT PROGRAM
C
C EXTERNAL DATASETS REQUIRED:
C      FOR S-MERGE : (1) UTLTYLBR.FOR  (GREEN)  [&XY3PRT.FOR]
C      NON S-MERGE : (2) LOOK.ASM      (ODG)
C=====
C*****PROGRAM XY3PRT
C-----
C 20 January 1984  Orville D. Green  DOS 2.0
C 28 April 1984  (odg) : modification to specify disk drives directly.
C
C XY3PRT prints the plots created by XY3PLT. The method is to read an
C input list of log files, retrieve the indicated plot file names
C from the log files, and then plot the files using a modified
C version of T. Engel's PDNPLT program.
C
C A maximum of 25 log files (100 plot files), and up to 100 additional
C plot files may be specified during a run.
C
C Link stream : LINK XY3PRT+LOOK;
C
C Subprogram references:
CALLS XY3GPF (0)
CALLS XY3PPF (0)
CALLS DXNUM (1)
C-----

```

PROGRAM XY3PRT

```

CHARACTER  LFNLST(25) *12, PFNLST(200) *12, BLANK *12,
1          PSTAT(2) *3, PF(4) *12, CHR *1
INTEGER    LFPLST(25), PFPLST(200), MAXNLF, NUMLFP, NUMPPF,

```

```

1          NPF,MAXNPF,DXLF,DXPF,DXNUM
LOGICAL    PRCFLG,ENDLFP,ENDPFP

```

```

C LFNLST = log filename list
C PFNLST = plot filename list
C LFPLST = log file processing list
C PFPLST = plot file processing list
C MAXNLF = maximum number of log files
C MAXNPF = maximum number of plot files
C NUMLFP = number of log files to process
C NumpFP = number of plot files to process
C DXLF   = disk drive containing log file(s)
C DXPF   = disk drive containing plot file(s)
C PRCFLG = processing flag. when .TRUE., indicates that the file lists
C         are not exhausted.
C ENDLFP = log file processing flag. When .TRUE. indicates that log
C         file list is exhausted.
C ENDPFP = plot file processing flag. When .TRUE. indicates that plot
C         file list is exhausted.

```

```

C      data statements
DATA BLANK /'          '/, PSTAT/'NOT','**'/
DATA LFPLST/25*0/, PFPLST/200*0/
DATA MAXNLF/25/,MAXNPF/200/,NUMLFP/0/,NumpFP/0/

```

```

C      format statements
91000 FORMAT('0XY3PRT - output XY3PLT created plots',/,
1         ' Input log file name(s) w/out drive spec (FILENAME.EXT)',
2         '/,1X,I2,' log files maximum, ¼CR½ when finished',/)
91001 FORMAT(1X,'Logfile # ',I2,' ? ',*,*)
91002 FORMAT(A12)
91040 FORMAT('0Log file disk drive letter (e.g., A) ? ',*,*)
91041 FORMAT(' Plot file disk drive letter (e.g., A) ? ',*,*)
91042 FORMAT(A1)
91020 FORMAT('0Input plot file name(s) w/out drive spec (FILENAME.EXT)',
1         ' ONLY for plot files NOT referenced in an above log file',
2         ' ¼CR½ terminates entry.')
91025 FORMAT(1X,'Plotfile # ',I2,' ? ',*,*)
95000 FORMAT('0**Log File Processing Status**',/)
95001 FORMAT(2X,'File : ',A12,1X,A3,1X,'processed')
96000 FORMAT('0**Plot File Processing Status**',/)

```

```

*Begin XY3PRT

```

```

C      input the list of log file names
1000 WRITE(*,91000) MAXNLF
DO 1010 I = 1, MAXNLF
WRITE(*,91001) I
READ (*,91002) LFNLST(I)
IF(LFNLST(I) .EQ. BLANK) GOTO 1020

```

```

1010  CONTINUE
1020  NUMLFP = I - 1

C      input list of individual plot files.
      WRITE(*,91020)
      DO 1030 I = 1, MAXNPF
          WRITE(*,91025) I
          READ (*,91002) PFNLST(I)
          IF(PFNLST(I) .EQ. BLANK) GOTO 1040
1030  CONTINUE
1040  NUPFPF = I - 1

C      input log/plot file disk drives
      WRITE(*,91040)
      READ (*,91042) CHR
      DXLF = DXNUM(CHR)
      WRITE(*,91041)
      READ (*,91042) CHR
      DXPF = DXNUM(CHR)

C      fetch the plot file names from the unprocessed, disk-resident
C      log files
2000  PRCFLG = .FALSE.
      DO 2020 I = 1, NUMLFP
          IF (LFPLST(I) .EQ. 0) THEN
              CALL XY3GPF(LFNLST(I),DXLF,PF,NPF)
              LFPLST(I) = NPF
C      put the plot files on the plot file processing list
              IF (NPF .NE. 0) THEN
                  DO 2010 J = 1, NPF
                      NUPFPF = NUPFPF + 1
                      PFNLST(NUPFPF) = PF(J)
2010  CONTINUE
                      PRCFLG = .TRUE.
                  ENDIF
              ENDIF
2020  CONTINUE

C      output the unprocessed, disk-resident plot files
3000  DO 3010 I = 1, NUPFPF
          IF(PFPLST(I) .EQ. 0) THEN
              CALL XY3PPF(PFNLST(I),PFPLST(I),DXPF)
              IF (PFPLST(I) .NE. 0) PRCFLG = .TRUE.
          ENDIF
3010  CONTINUE

C      the user may have exchanged diskettes during processing.
C      continue processing if this is the case
4000  IF (PRCFLG) GOTO 2000

```

```

C      output the log file processing status
5000  ENDLFP = .TRUE.
      WRITE(*,95000)
      DO 5100 I = 1, NUMLFP
        IF (LFPLST(I) .EQ. 0) J = 1
        IF (LFPLST(I) .NE. 0) J = 2
        WRITE(*,95001) LFNLSST(I),PSTAT(J)
        IF (J .EQ. 1) ENDLFP = .FALSE.
5100  CONTINUE

C      output the plot file processing status
6000  ENDPFP = .TRUE.
      WRITE(*,96000)
      DO 6100 I = 1, NUMPFP
        IF (PFPLST(I) .EQ. 0) J = 1
        IF (PFPLST(I) .NE. 0) J = 2
        WRITE(*,95001) PFNLSST(I),PSTAT(J)
        IF (J .EQ. 1) ENDPFP = .FALSE.
6100  CONTINUE

C      continue processing ?
7000  IF ( (.NOT. ENDPFP) .OR. (.NOT. ENDLFP) ) THEN
      WRITE(*,'(1X,1H) ')
      PAUSE 'Exchange diskette(s) to continue processing'
      GOTO 2000
    ENDIF

```

```

*Terminate XY3PRT
STOP '***XY3PRT processing request(s) completed'
END

```

```

C-----
C*****SUBROUTINE XY3GPF(LF,DXLF,PF,NPF)
C-----
C 20 January 1984 Orville D. Green IBM/PC DOS 2.0
C 24 April 1984 (odg) : modification to specify disk drives directly.

```

C Description:

C XY3GPF processes a log file to get the plot files specified.

C Arguments:

C	LF	[Character *12]	:Log file name
C	DXLF	[Integer]	:Log file disk number (1=A,2=B,etc)
C	PF	[Character *12]	:List of plot file names
C	NPF	[Integer]	:Number of plot files

C Subprogram references:

CALLS SEARCH (1)

C-----

SUBROUTINE XY3GPF(LF,DXLF,PF,NPF)

\$DEBUG

C declarations

CHARACTER LF *12, PF(4) *12, DRIVE(9) *2, LOGFIL *14
INTEGER DXLF,NPF,LFHRB(30),FOUND

C data statements

DATA DRIVE /'A:', 'B:', 'C:', 'D:', 'E:', 'F:', 'G:', 'H:', 'I:' /

*Begin XY3GPF

NPF = 0

C See if the log file is currently disk-resident.
CALL SEARCH (LF,DXLF,FOUND)

C if it is, get the plot file names
IF (FOUND .EQ. 0) THEN

C first, open the log file

WRITE(LOGFIL, '(A2,A12)') DRIVE(DXLF),LF
WRITE(*,80000) LOGFIL

80000 FORMAT('0Processing logfile : ',A14,1X,'on unit 7')
OPEN(7,FILE=LOGFIL,STATUS='OLD',ACCESS='DIRECT',
1 FORM='FORMATTED',RECL=60)

C read the header record to get the number of plot files.
READ(7,'(30I2)',REC=1) LFHRB
NPF = 0

IF (LFHRB(2) .NE. 0) NPF = NPF + 1
IF (LFHRB(9) .NE. 0) NPF = NPF + 1
IF (LFHRB(15) .NE. 0) NPF = NPF + 1
IF (LFHRB(22) .NE. 0) NPF = NPF + 1

C read the plot file names

DO 100 I = 1, NPF
READ(7,'(A12)',REC=13*(I-1)+2) PF(I)

100 CONTINUE

ENDIF

*Terminate XY3GPF

CLOSE(7,STATUS='KEEP')
RETURN
END

C-----
C*****SUBROUTINE XY3PPF(PF,PL,DXPF)
C-----

C 20 January 1984 Orville D. Green IBM/PC DOS 2.0

C 24 April 1984 (odg) : modification to specify disk drive directly.

C Description:

C XY3PPF plots the indicated plot file

C Arguments:

C PF	[character *12]	:plot file name
C PL	[integer]	:plot file status flag
C DXPF	[integer]	:plot file disk number (1=A,2=B,etc)

C Subprogram references:

CALLS SEARCH (1)
CALLS PDNPL (0)

SUBROUTINE XY3PPF(PF,PL,DXPF)

\$DEBUG

C declarations

CHARACTER	PF *12, PFN *14, DRIVE(9) *2
INTEGER	PL, DXPF

C data statements

DATA DRIVE /'A:', 'B:', 'C:', 'D:', 'E:', 'F:', 'G:', 'H:', 'I:' /

*Begin XY3PPF

C is the plot file currently disk resident ?
CALL SEARCH(PF,DXPF,PL)

C if so, plot it
IF (PL .EQ. 0) THEN
WRITE(PFN, '(A2,A12)') DRIVE(DXPF), PF
CALL PDNPL(PFN)
ENDIF

*Terminate XY3PPF

RETURN
END

C-----
C*****SUBROUTINE PDNPL(PPDN)

C-----
C ***T. Engel PDNPLT program modified for subroutine call by ODGreen***
C ***PDNPLT Version 1.02 - 26 May 1983 by Tom Engel.

C PPDN = plot file specification

SUBROUTINE PDNPL(PPDN)
CHARACTER *14 PPDN,DPDN
CHARACTER *1 CA(820),BS

```

      INTEGER      UNIT
      DATA UNIT/99/

```

```

1002 FORMAT (' PDNPLOT - Dataset ',A14,' OPENed on unit',I3)
1003 FORMAT (' PDNPLOT - Dataset ',A14,' closed, status=keep.')
6000 FORMAT ('1',3A1)
6001 FORMAT (' ',820A1)
6002 FORMAT (' Plot complete.',/, '+')

```

```

      WRITE(*,'(A)') ' PDNPLOT Version 1.02. 26 May 83 '
      WRITE(*,'(A)') ' Copyright - Tom Engel'

```

```

      IUN = 0
      BS=CHAR(8)
      IF (PPDN.EQ.' ') GO TO 100
      CA(1)=CHAR(27)
      CA(2)='A'
      CA(3)=CHAR(8)
      WRITE(*,'(A)') ' PDNPLOT - Device LPT1: OPENed on unit 6.'
      OPEN (6,FILE='LPT1:')
      WRITE(*,1002) PPDN,UNIT
      OPEN(UNIT,FILE=PPDN,STATUS='OLD',ACCESS='DIRECT',FORM=
1      'UNFORMATTED',RECL=820)
      WRITE(6,6000) (CA(I),I=1,3)
      DO 40 I=2,95
      READ(UNIT,REC=I) CA
      WRITE(6,6001) CA
40  CONTINUE
      WRITE(*,6002)
      WRITE(*,1003) PPDN
      CLOSE(UNIT,STATUS='KEEP')
      WRITE(*,'(A)') ' PDNPLOT - Device LPT1: closed.'
      CLOSE(6)
100  WRITE(*,'(A)') ' PDNPLOT - Normal exit.'
      RETURN
      END

```

Three plots per page

```

C=====
C SOURCE FILE: XY3PLT.FOR  **COMPILE USING MS-FORTRAN 3.04**
C
C 02 MARCH 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
C
C LOCALLY CONTAINED PROGRAM UNITS:
C [S]  XY3PLT : BUILDS 3 XY PLOTS PER 8.5 X 11 INCH PAGE
C [S]  XY3LOG : PROCESSES LOG FILE INFORMATION
C [S]  XY3SW  : PARTITIONS PAGE AND SETS WINDOWS
C [S]  XY3DAT : OUTPUTS PLOT RUN DATE

```

C

```

C          IP(3) = # tick marks on x scale [24, *]
C          IP(4) = frequency of x labels [2]
C          IP(5) - IP(8) = y specs [3],[8],[10],[1]
C          IP(9) = size of title label [8]
C          IP(10) = size of tick mark [4]
C          * XXP  $\frac{1}{2}$  24, then XXP rounded up to nearest
C          integer is used

```

C Subprogram references :

```

CALLS XY3LOG      (0)
CALLS SCAXLR      (1)
CALLS INITT       (4)
CALLS XY3SW       (0)
CALLS XY3DAT      (0)
CALLS DRAXLR      (1)
CALLS PLOTXY      (1)
CALLS LBAXLR      (1)
CALLS XY3PTT      (1)
CALLS PFINIS      (4)
CALLS XY3CLG      (0)

```

```

C-----
C          SUBROUTINE XY3PLT (N,X,Y,VM,XSN,XSX,YSN,YSX,LF,LU,LD,LP,TX,IP)

```

C DECLARATIONS

```

      INTEGER      IP(10)
      REAL         X(1), Y(1)
      CHARACTER     LF *14, TX(3,15) *4

```

```

C
C IPD = plotting parameter defaults
C IPP = plotting parameters :
C      I = abscissa, J = ordinate
C      -NDSL = # digits in scale label
C      -SSL = size of scale label
C      -NTS = target # ticks on axis
C      -FTL = frequency of scale labels
C      STL = size of title label
C      TICK = size of tick mark
C

```

```

      INTEGER      IPD(10),IPP(10),STL,TICK
      EQUIVALENCE  (IPP(1),INDSL), (IPP(2),ISSL), (IPP(3),INTS),
1      (IPP(4),IFTL), (IPP(5),JNDSL), (IPP(6),JSSL),
2      (IPP(7),JNTS), (IPP(8),JFTL), (IPP(9), STL),
3      (IPP(10),TICK)

```

C Scaled window boundaries :

```

C 1st character : X =  $\frac{1}{2}$  abscissa, Y =  $\frac{1}{2}$  ordinate
C 2nd character : X =  $\frac{1}{2}$  maximum, N =  $\frac{1}{2}$  minimum
C 3rd character : W =  $\frac{1}{2}$  window, P =  $\frac{1}{2}$  plot
C e.g., XNW =  $\frac{1}{2}$  scaled minimum x window boundary

```

C WINDOW = current window number

INTEGER WINDOW

C Log/Plot file information

C LFHRN = log file header record number
 C LFHRB = log file header record buffer
 C PFHRN = plot file header record number
 C PFHRB = plot file header record buffer
 C TXn = character *1 encoding of TX
 C OPF = old plot file
 C PF = current plot file

INTEGER	LFHRB(30), PFHRN
CHARACTER *1	PFHRB(60), TX1(60), TX2(60), TX3(60)
CHARACTER *14	OPF, PF

C Scaling information

C ISF = abscissa scaling factor
 C JSF = ordinate scaling factor
 C XSS = abscissa scale step size
 C YSS = ordinate scale step size
 C INS = number of steps on abscissa
 C JNS = number of steps on ordinate

C Data statements

DATA IPD /2,8,24,2,3,8,10,1,8,4/

C Format statements

91020 FORMAT(4A1)

*Begin XY3PLT

C initialize plotting parameters

```
1000 DO 1010 I = 1, 10
      IF (IP(I) .LE. 0) THEN
        IPP(I) = IPD(I)
      ELSE
        IPP(I) = IP(I)
      ENDIF
1010 CONTINUE
      INTST = INT(XSX)
      IF ( (XSX - REAL(INTST)) .NE. 0.0) INTST = INTST + 1
      INTS = MAX0(INTS, INTST)
```

C initialize the log/plot file information

```
K = 1
DO 1020 I = 1, 60
  PFHRB(I) = ' '
  IF (I .LE. 15) THEN
```

```

        READ(TX(1,I),91020) (TX1(J),J=K,K+3)
        READ(TX(2,I),91020) (TX2(J),J=K,K+3)
        READ(TX(3,I),91020) (TX3(J),J=K,K+3)
        K = K + 4
    ENDIF
1020 CONTINUE

C      Process the log file
      CALL XY3LOG (LF,LU,LD,LP,PF,LFHRN,PFHRN,LFHRB,PFHRB,WINDOW)

C      Set up the coordinate axis scales
      XNP = XSN
      XXP = XSX
      YNP = YSN
      YXP = YSX
      CALL SCAXLR (N,X,VM,XNP,XXP,INTS,INDSL,XSS,INS,ISF)
      CALL SCAXLR (N,Y,VM,YNP,YXP,JNTS,JNSDL,YSS,JNS,JSF)

C      Initialize plotting library
      IF (WINDOW .EQ. 1) THEN
          OPF = ' '
      ELSE
          OPF = PF
      ENDIF
      CALL INITT (OPF,PF)

C      Output the run date, if this is a new plot
      IF (WINDOW .EQ. 1) THEN
          CALL XY3SW (0,XNP,XXP,YNP,YXP,XNW,XXW,YNW,YXW)
          CALL XY3DAT (XNW,YNW)
      ENDIF

C      Set the windows for the current plot
      CALL XY3SW (WINDOW,XNP,XXP,YNP,YXP,XNW,XXW,YNW,YXW)

C      Draw the axes
      CALL DRAXLR (YNP,XNP,XSS,INS,TICK,1)
      CALL DRAXLR (YXP,XNP,XSS,INS,TICK,1)
      CALL DRAXLR (XNP,YNP,YSS,JNS,TICK,2)
      CALL DRAXLR (XXP,YNP,YSS,JNS,TICK,2)

C      Plot the data
      CALL PLOTXY (N,X,Y,VM,0)

C      Label the axes
      CALL LBAXLR (XNP,XSS,YNP,0.0,INS,ISF,INDSL,ISSL,TICK,IFTL,1)
      CALL LBAXLR (XNP,0.0,YNP,YSS,JNS,JSF,JNSDL,JSSL,TICK,JFTL,1)

C      Output the titles
      CALL XY3PTT (XNW,YNP,TX1,TX2,TX3,JSF,ISSL,TICK,STL,1)

```

C Close the plot file
CALL PFINIS ('KEEP')

C Update and close the log file
CALL XY3CLG (LF,LU,TX1,TX2,TX3,WINDOW,LFHRN,LFHRB,PFHRN,PFHRB)

*Terminate XY3PLT

99999 RETURN
END

C-----
C*****SUBROUTINE XY3LOG (LF,LU,LD,LP,PF,LFHRN,PFHRN,LFHRB,PFHRB,WINDOW)
C-----

C 10 October 1983 Orville D. Green IBM/PC DOS 2.0
C 20 March 1984 (odg) : modified to create new LF when old LF is full

C Description:

C XY3LOG accesses the log file (or creates and initializes a new log
C file) to obtain information about the plot file targeted for
C processing. XY3LOG also updates the header record buffers for the
C current plot. (These records are not written to file -- this is
C done by XY3CLG after a successful plot run).

C The log file contains 53 (60 byte) records containing information on
C up to 4 plot files. [Maximum number of plot files that may be
C maintained on a double-sided, double-density diskette]. The log
C file structure is as follows :

C Record 1 : log file header record.
C Organized into 4 "plot file" blocks :
C Block 1: (bytes 1-14) - Plot file #1 block
C bytes 1 - 2 : 1 if plot file full, 0 otherwise
C bytes 3 - 4 : 1 if window 1 full, 0 otherwise
C bytes 5 - 6 : 1 if window 2 full, 0 otherwise
C bytes 7 - 8 : 1 if window 3 full, 0 otherwise
C bytes 9 - 10 : 1 if window 4 full, 0 otherwise
C bytes 11 - 12 : 1 if window 5 full, 0 otherwise
C bytes 13 - 14 : 1 if window 6 full, 0 otherwise
C Block 2: (bytes 15-28) - Plot file #2 block
C Block 3: (bytes 29-42) - Plot file #3 block
C Block 4: (bytes 43-56) - Plot file #4 block
C note: (bytes 57-60) not used

C Records 2 - 53 : Plot file segments 1 - 4.
C Record 2 - 14 : Plot file segment #1
C Record 2 : Plot file header record. *
C Records 3-5 : Window 1 titles
C Records 6-8 : Window 2 titles

```

C      Records 9-11 : Window 3 titles
C      Records 12  : Window 4 title
C      Record 13   : Window 5 title
C      Record 14   : Window 6 title
C      Records 15-27 : Plot file segment #2
C      Records 28-40 : Plot file segment #3
C      Records 41-53 : Plot file segment #4

C      *Plot file header record structure :
C      Bytes 1-12 : Plot filename specification
C      Bytes 13-20 : Date/time window 1 built
C      Bytes 21-28 : Date/time window 2 built
C      Bytes 29-36 : Date/time window 3 built
C      Bytes 37-44 : Date/time window 4 built
C      Bytes 45-52 : Date/time window 5 built
C      Bytes 53-60 : Date/time window 6 built

C Arguments:
C LF      [character *14]      : log filename specification
C LU      [integer]           : log file logical unit
C LD      [integer]           : log file drive number
C LP      [integer]           : pause indicator (0 = 1/2 pause)
C LFHRN   [integer]           : log file header record number
C PFHRN   [integer]           : plot file header record number
C LFHRB   [integer]           : log file header record buffer (30)
C PFHRB   [character *1]      : plot file header record buffer (60)
C WINDOW  [integer]           : window number

C Subprogram references:
CALLS BFSDAT      (2)
C-----
C
      SUBROUTINE XY3LOG (LF,LU,LD,LP,PF,LFHRN,PFHRN,LFHRB,PFHRB,WINDOW)

C      declarations
      CHARACTER      LF*14,PF*14,PFHRB(60)*1,      FN*8,FX*3,FS*14,FD*2
      INTEGER        PFHRN,LFHRB(30),WINDOW
      LOGICAL        VALHRB

C VALHRB = .TRUE. if valid header record block

C      format statements
91000 FORMAT('0*XY3LOG* FILE:',1X,A14,1X,'UNIT:',1X,I4,1X,'OPENED')
91010 FORMAT(30I2)
91020 FORMAT(60A1)
92010 FORMAT('0*XY3LOG* FILE:',1X,A14,1X,'UNIT:',1X,I4,1X,'CLOSED')
93000 FORMAT(2X,12A1)
93001 FORMAT(4X,8A1)
93002 FORMAT(A2)
93003 FORMAT(A2,12A1)

```

*Begin XY3LOG

C Open the log file.

```
1000 IF (LF .NE. ' ') THEN
      OPEN(LU,FILE=LF,STATUS='OLD',ACCESS='DIRECT',
1       FORM='FORMATTED',RECL=60)
      WRITE(*,91000) LF, LU
```

C Create a log file
ELSE

C Pause for diskette exchange, if indicated
IF (LP .EQ. 0) THEN
 PAUSE ' XY3PLT log file creation. Insert diskette'
ENDIF

C Build the new log file spec
CALL BFSDAT (LD,'XY','YM','DD','HH','L','NN',FN,FX,LF)

C Open the log file
OPEN(LU,FILE=LF,STATUS='NEW',ACCESS='DIRECT',
1 FORM='FORMATTED',RECL=60)
WRITE(*,91000) LF, LU

C Initialize the log file
LFHRN = 1
DO 1010 I = 1, 30
 LFHRB(I) = 0
1010 CONTINUE
 WRITE(LU,91010,REC=LFHRN) LFHRB
 DO 1020 I = 2, 53
 WRITE(LU,91020,REC=I) PFHRB
1020 CONTINUE
ENDIF

C Get the plot file block from the log file header
2000 READ(LU,91010,REC=LFHRN) LFHRB
 DO 2010 IPFB = 1, 22, 7
 IF (LFHRB(IPFB) .EQ. 0) GOTO 2020
 VALHRB = LFHRB(IPFB) .EQ. 1
2010 CONTINUE

C File full ? Open a new one. Else, abnormal ending
IF (.NOT. VALHRB) THEN
 STOP 'ABEND XY3PLT ABORTED. XY3LOG:invalid logfile header block'
ELSE
 CLOSE(LU)
 WRITE(*,92010) LF,LU
 LF = ' '

```

        GOTO 1000
    ENDIF

C      Get the target window from the plot file block
2020  DO 2030 ITW = IPFB + 1, IPFB + 3
        IF (LFHRB(ITW) .EQ. 0) GOTO 2040
2030  CONTINUE
        STOP 'ABEND XY3PLT ABORTED. XY3LOG:invalid log file window field'

C      Calculate the window and plot file header record numbers
2040  WINDOW = ITW - IPFB
        PFHRN = (IPFB/7) * 13 + 2

C      Build a Plot file spec
        CALL BFSDAT (LD,'XY','YM','DD','HH','P','NN',FN,FX,FS)

C      Update the plot file header buffer. For window 1, a new plot file
C      is built. Otherwise, the plot file is obtained from the plot file
C      header buffer.
3000  IF (WINDOW .EQ. 1) THEN
        PF = FS
        READ(FS,93000) (PFHRB(J),J=1,12)
        READ(FS,93001) (PFHRB(J),J=13,20)
    ELSE
        READ (LU,91020,REC=PFHRN) PFHRB
        READ(FS,93002) FD
        WRITE(PF,93003) FD,(PFHRB(J),J=1,12)
        I = (WINDOW - 1) * 8 + 13
        READ (FS,93001) (PFHRB(J),J=I,I+7)
    ENDIF

C      Update the log file header buffer
        LFHRB(ITW) = 1
        IF (WINDOW .EQ. 3) THEN
            LFHRB(IPFB) = 1
        ENDIF

*Terminate XY3LOG

99999 RETURN
    END

-----
C*****SUBROUTINE XY3SW (WINDOW,XNP,XXP,YNP,YXP,XNW,XXW,YNW,YXW)
-----
C 10 October 1983 Orville D. Green IBM/PC DOS 2.0

C Description:
C XY3SW partitions the page into 4 windows, and sets the windows
C for the current plot. Window 0 corresponds to the date window.

```

C Arguments:

C WINDOW	[integer]	: window number
C XNP	[real]	: minimum scaled abscissa plot boundary
C XXP	[real]	: maximum scaled abscissa plot boundary
C YNP	[real]	: minimum scaled ordinate plot boundary
C YXP	[real]	: maximum scaled ordinate plot boundary
C XNW	[real]	: minimum scaled abscissa window boundary
C XXW	[real]	: maximum scaled abscissa window boundary
C YNW	[real]	: minimum scaled ordinate window boundary
C YXW	[real]	: maximum scaled ordinate window boundary

C Subprogram references:

CALLS SCLWIN (1)
 CALLS TWINDO (4)
 CALLS DWINDO (4)

C-----

SUBROUTINE XY3SW (WINDOW,XNP,XXP,YNP,YXP,XNW,XXW,YNW,YXW)

C declarations

INTEGER	WINDOW
INTEGER	INW(4),IXW(4),JNW(4),JXW(4),
1	INP(4),IXP(4),JNP(4),JXP(4)

C data statements

DATA INW	/ 65, 20, 20, 20/
DATA IXW	/460,460,460,460/
DATA JNW	/610,410,215, 20/
DATA JXW	/623,605,410,215/
DATA INP	/ 65, 65, 65, 65/
DATA IXP	/460,410,410,410/
DATA JNP	/610,455,260, 65/
DATA JXP	/623,590,395,200/

*Begin XY3SW

C Determine the scaled window boundaries

NW = WINDOW + 1
 CALL SCLWIN (INP(NW),IXP(NW),INW(NW),IXW(NW),XNP,XXP,XNW,XXW)
 CALL SCLWIN (JNP(NW),JXP(NW),JNW(NW),JXW(NW),YNP,YXP,YNW,YXW)

C Set the windows

CALL TWINDO (INW(NW),IXW(NW),JNW(NW),JXW(NW))
 CALL DWINDO (XNW,XXW,YNW,YXW)

*Terminate XY3SW

99999 RETURN
 END

C-----

C*****SUBROUTINE XY3DAT (XNW,YNW)

C-----

C 10 October 1983 Orville D. Green IBM/PC DOS 2.0

C Description:

C XY3DAT outputs the initial plot run date

C Arguments:

C XNW [real] : minimum scaled abscissa window boundary

C YNW [real] : minimum scaled ordinate window boundary

C Subprogram references:

CCALLS GDATE (5)

CALLS NCODI1 (2)

CALLS MOVEA (4)

CALLS BPOSN (1)

CALLS MOVABS (4)

CALLS MESSAGE (3)

C-----

SUBROUTINE XY3DAT (XNW,YNW)

C declarations

CHARACTER K*56,CM(12)*3,C(56)*1,CV(11)*1

C data statements

DATA CM / 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
1 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec' /

*Begin XY3DAT

K = '

Run Date : '

READ(K, '(56A1)') (C(J), J=1, 56)

CALL GDATE (IYR, IMON, IDAY)

CALL NCODI1(IDAY, CV)

C(46) = CV(10)

C(47) = CV(11)

READ(CM(IMON), '(3A1)') (C(J), J=49, 51)

CALL NCODI1(IYR, CV)

C(53) = CV(8)

C(54) = CV(9)

C(55) = CV(10)

C(56) = CV(11)

CALL MOVEA (XNW, YNW)

CALL BPOSN (IX, IY)

IY = IY + 5

CALL MOVABS (IX, IY)

CALL MESSAGE (IX, IY, C, 56, 8, 0.0, 1)

*Terminate XY3DAT

99999 RETURN

END

C-----
C*****SUBROUTINE XY3PTT (XNW,YNP, TX1, TX2, TX3, JSF, ISSL, TICK, STL, IPR)
C-----

C 10 October 1983 Orville D. Green IBM/PC DOS 2.0

C Description:

C XY3PTT outputs the titles for the plot.

C Arguments:

C	XNW	[real]	:	minimum scaled abscissa window boundary
C	YNP	[real]	:	minimum scaled ordinate plot boundary
C	TX1	[character *1]	:	first abscissa title line (60)
C	TX2	[character *1]	:	second abscissa title line (60)
C	TX3	[character *1]	:	third abscissa title line (60)
C	JSF	[integer]	:	scaling factor for ordinate
C	ISSL	[integer]	:	size of abscissa scale label
C	TICK	[integer]	:	size of tick mark
C	STL	[integer]	:	size of title label
C	IPR	[integer]	:	position indicator -
C				1/20 below, left
C				1/40 above, right

C Subprogram references:

CALLS MOVEA (4)
CALLS BPOSN (1)
CALLS MOVABS (4)
CALLS MESSAGE (3)
CALLS NCODI1 (2)

C-----
SUBROUTINE XY3PTT (XNW,YNP, TX1, TX2, TX3, JSF, ISSL, TICK, STL, IPR)

C declarations

INTEGER	TICK, STL
CHARACTER	TX1(60)*1, TX2(60)*1, TX3(60)*1, TY(20)*1,
1	CS(11)*1, TY0*20

C data statements

DATA TY0 /' Scale = 10 ** '/

*Begin XY3PTT

C Label the abscissa

CALL MOVEA (XNW, YNP)

CALL BPOSN (IX, IY)

IY = IY - IPR * (TICK + ISSL + STL + 2)

```

CALL MOVABS (IX,IY)
CALL MESSAGE (IX,IY,TX1,60,STL,0.0,1)
IY = IY - IPR * (STL + 2)
CALL MOVABS (IX,IY)
CALL MESSAGE (IX,IY,TX2,60,STL,0.0,1)
IY = IY - IPR * (STL + 2)
CALL MOVABS (IX,IY)
CALL MESSAGE (IX,IY,TX3,60,STL,0.0,1)

C   Label the ordinate
READ (TY0,'(20A1)') (TY(J),J=1,20)
CALL NCOD11 (JSF,CS)
J = 9
CS (J) = CS(1)
DO 30 I = 17, 19
    TY(I) = CS(J)
    J = J + 1
30  CONTINUE
CALL MOVEA (XNW,YNP)
CALL BPOSN (IX,IY)
IX = IX + STL
CALL MOVABS (IX,IY)
CALL MESSAGE (IX,IY,TY,20,STL,90.0,1)

*Terminate XY3PTT

99999 RETURN
END

C-----
C*****SUBROUTINE XY3CLG(LF,LU,TX1,TX2,TX3,WNDOW,LFHRN,LFHRB,PFHRN,PFHRB)
C-----
C 10 October 1983  Orville D. Green  IBM/PC  DOS 2.0

C Description :
C   XY3CLG writes the updated log file header records to file, and
C   closes the log file.

C Arguments :
C   LF      [character *14]      : log filename specification
C   LU      [integer]            : log file logical unit
C   TX1     [character *1]       : first abscissa title line (60)
C   TX2     [character *1]       : second abscissa title line (60)
C   TX3     [character *1]       : third abscissa title line (60)
C   WINDOW  [integer]            : window number
C   LFHRN   [integer]            : log file header record number
C   LFHRB   [integer]            : log file header record buffer
C   PFHRN   [integer]            : plot file header record number
C   PFHRB   [integer]            : plot file header record buffer
C-----

```

```

SUBROUTINE XY3CLG(LF,LU,TX1,TX2,TX3,WNDOW,LFHRN,LFHRB,PFHRN,PFHRB)

C   declarations
CHARACTER      LF*14, TX1(60)*1, TX2(60)*1, TX3(60)*1, PFHRB(60)*1
INTEGER        WNDOW, LFHRB(30), PFHRN

C   format statements
91000 FORMAT (30I2)
91001 FORMAT (60A1)
91002 FORMAT('0*XY3CLG* FILE:',1X,A14,1X,'UNIT:',1X,I4,1X,'CLOSED')

*Begin XY3CLG

C   write out the header records
1010 WRITE(LU,91000,REC=LFHRN) LFHRB
    WRITE(LU,91001,REC=PFHRN) PFHRB

C   write out the abscissa titles
ITREC = PFHRN + (WNDOW - 1) * 3 + 1
WRITE (LU,91001,REC=ITREC) TX1
ITREC = ITREC + 1
WRITE (LU,91001,REC=ITREC) TX2
ITREC = ITREC + 1
WRITE (LU,91001,REC=ITREC) TX3

C   close the log file
CLOSE (LU)
WRITE (*,91002) LF, LU

*Terminate XY3CLG

99999 RETURN
END

```

One plot per page

```

C=====
C SOURCE FILE:  XYPLOT.FOR
C
C 16 JUNE 1984  ORVILLE D. GREEN  IBM/PC  MS FORTRAN 3.20  DOS 2.0
C
C LOCALLY CONTAINED PROGRAM UNITS:
C [S]  XYPLOT   :  BUILD ONE XY PLOT PER PAGE
C [S]  XYPDAT   :  BUILD XYPLOT DATE WINDOW
C
C EXTERNAL DATASETS REQUIRED:
C      FOR S-MERGE :  (1) XYPLTLBR.FOR  (GREEN)  [S-MERGE COPY]
C                  (2) UTLTYLBR.FOR  (GREEN)  [&XYPLOT.FOR]
C
C                  (2) UTLTYLBR.FOR  (GREEN)  [&XYPLOT.FOR]

```

```

C          (3)  UTILIB.FOR      (ENGEL)      [$XYPLOT.FOR]
C          (4)  PLIBMEM.FOR     (ENGEL)      [$XYPLOT.FOR]
C      NON S-MERGE : (5)  GDATE.ASM    (GREEN)
C          (6)  REMOVE.ASM      (GREEN)
C=====
C*****SUBROUTINE XYPLOT (N,X,Y,VM,XN,XX,YN,YX,OPF,PF,IP,TX)
C-----
C 16 JUNE 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
C      XYPLOT UTILIZES TOM ENGEL'S PLOT-10 EMULATION PACKAGE TO BUILD
C      ONE LINEAR XY PLOT.  THE PLOT MAY BE EITHER A "LINE PLOT",
C      WITH XY POINTS CONNECTED, OR A "POINT PLOT" (SCATTER DIAGRAM)
C      WITH XY POINTS ONLY PLOTTED, OR BOTH.
C      THE DEFAULT PLOT PAGE IS 8.5 X 11 INCHES, WITH MARGINS SET
C      ACCORDING TO THE "GUIDE FOR PREPARATION OF THESES AND PRO-
C      FESSIOANAL PAPERS", COLLEGE OF GRADUATE STUDIES, MONTANA STATE
C      UNIVERSITY, BOZEMAN, MONTANA 59717.
C      THE DEFAULT PLOT PAGE MAY BE OVERWRITTEN WITH CALLER ARGUMENTS.
C      THE PLOT IS WRITTEN TO FILE.  THE CALLER MAY REQUEST A HARDCOPY
C      AND SAVE OR DELETE THE PLOT FILE.
C
C LOGICAL UNIT ASSIGNMENTS
C      AS DEFINED IN PLIBMEM (-99,-98,ETC.) -- NONE USED LOCALLY
C
C ENTRY
C      N [I]  NUMBER OF (X,Y) PAIRS
C      X [R]  ARRAY OF X (ABSCISSA) VALUES
C      Y [R]  ARRAY OF Y (ORDINATE) VALUES
C      VM [R]  VALUE ASSIGNED TO A MISSING DATUM
C              ** MISSING DATA ARE NOT PLOTTED
C      XN [R]  MINIMUM VALUE ON ABSCISSA SCALE
C      XX [R]  MAXIMUM VALUE ON ABSCISSA SCALE
C      YN [R]  MINIMUM VALUE ON ORDINATE SCALE
C      YX [R]  MAXIMUM VALUE ON ORDINATE SCALE
C              ** THE ACTUAL SCALE USED ON THE PLOT MAY DIFFER FROM THE
C              PASSED SCALE IF THE DATA FALL OUTSIDE THE PASSED
C              LIMITS, OR THE SCALE COMES OUT UNEVEN.  THESE ARGU-
C              MENTS ALLOW THE CALLER TO MAINTAIN A CONSISTENT SCALE
C              FROM PLOT TO PLOT, WHICH AUTOSCALING MAY NOT PROVIDE.
C              ** AUTOSCALING MAY BE INVOKED BY SWITCHING THE MIN AND
C              MAX SCALE VALUES
C      OPF [C*14]  OLD PLOT FILE SPECIFICATION, OR ' ' IF NEW PLOT FILE
C              ** ALLOWS MORE THAN ONE PLOT PER PAGE, BUT ABSCISSA
C              AND ORDINATE SCALES SHOULD BE IDENTICAL AMONG THE
C              PLOTS.
C      PF [C*14]  PLOT FILE SPECIFICATION
C      IP [I]  PLOTTING PARAMETERS (44 ELEMENTS)
C              ** PASSING IP(I) = 0 ENABLES DEFAULT []
C              ** PASSING IP(I) = -1 SETS IP(I) = 0 LOCALLY
C              ** SIZES SPECIFIED IN EPSON PLOT UNITS
C              (815 X 623 = 15" X 11")

```

C*****RUNDATE BANNER

C-----THE RUNDATE BANNER IS A SINGLE LINE OF TEXT DEFAULTED
 C FOR OUTPUT AT THE TOP OF THE PLOT. THE DEFAULT RUNDATE
 C BANNER IS A 40 CHARACTER STRING READING :
 C "XYPLOT(840616) RUN DATE : DD MON YYYY"
 C IT MAY BE DISABLED OR OVERWRITTEN WITH A CALLER PASSED
 C BANNER. NOTE THAT BY ENABLING THE CALLER RUNDATE
 C BANNER, AND CHANGING THE WINDOW BOUNDARIES, THE CALLER
 C MAY OUTPUT A SINGLE LINE OF TEXT ANYWHERE ON THE OUTPUT
 C PAGE.

C IP(1) RDO = 0 ENABLE DEFAULT RUNDATE BANNER []
 C = 1 DISABLE RUNDATE BANNER
 C = 2 ENABLE CALLER RUNDATE BANNER

C IP(2) RDOZ SIZE OF RUNDATE BANNER [8]
 C IP(3) RDOL LENGTH OF CALLER RUNDATE BANNER [0]

C-----RUNDATE BANNER WINDOW

C IP(4) INWD MIN X ABS WINDOW BOUNDARY [85]
 C IP(5) IXWD MAX X ABS WINDOW BOUNDARY [405]
 C IP(6) JNWD MIN Y ABS WINDOW BOUNDARY [500]
 C IP(7) JXWD MAX Y ABS WINDOW BOUNDARY [510]

C*****XY PLOTS

C-----TWO TYPES OF XY PLOTS ARE AVAILABLE: A LINE PLOT AND A
 C POINT PLOT. EITHER, BOTH, OR NEITHER MAY BE REQUESTED.
 C THE DEFAULT IS BOTH.

C IP(8) LPO = 0 ENABLE LINE PLOT []
 C = 1 DISABLE LINE PLOT
 C IP(9) LT LINE PLOT DASH TYPE SPECIFICATION [0]
 C IP(10) PPO = 0 ENABLE POINT PLOT []
 C = 1 DISABLE POINT PLOT

C IP(11) PPC POINT PLOT CHARACTER (ASCII) [43]
 C IP(12) PPCZ SIZE OF POINT PLOT CHAR [10]

C-----WHEN A PREVIOUSLY CONSTRUCTED PLOT IS BEING OVERLAYED
 C DURING THE CURRENT CALL, IT MAY NOT BE NECESSARY TO
 C REDRAW THE AXES

C IP(13) AXES = 0 ENABLE AXES DRAW []
 C = 1 DISABLE AXES DRAW

C IP(14) TICK SIZE OF AXES TICK MARK [4]

C-----AXES LABELLING CONSISTS OF DEFINING THE NUMBER OF DIGITS
 C TO BE DISPLAYED BY THE TICK MARKS, THE FREQUENCIES AT
 C WHICH THEY ARE TO BE WRITTEN (E.G., 2 MEANS EVERY OTHER
 C TICK MARK), AND AN OPTIONAL PARAMETER LABEL.
 C THE PARAMETER LABELS MAY BE DISABLED BY SPECIFYING A
 C LENGTH OF MINUS 1.

C-----ABSCISSA

C IP(15) NDX NUMBER OF DIGITS IN X SCALE LABEL [2]
 C IP(16) DXZ SIZE OF X SCALE LABEL [8]
 C IP(17) NTX NUMBER OF TICK MARKS ON X SCALE [10]
 C IP(18) FTX FREQUENCY OF X SCALE LABELS [1]
 C IP(19) XPL LENGTH OF X PARAMETER LABEL [30]
 C IP(20) XPLZ SIZE OF X PARAMETER LABEL [8]

```

C-----ORDINATE
C      IP(21) NDY      NUMBER OF DIGITS IN Y SCALE LABEL [3]
C      IP(22) DYZ      SIZE OF Y SCALE LABEL [8]
C      IP(23) NTY      NUMBER OF TICK MARKS ON Y SCALE [10]
C      IP(24) FTY      FREQUENCY OF Y SCALE LABELS [1]
C      IP(25) YPL      LENGTH OF Y PARAMETER LABEL [30]
C      IP(26) YPLZ     SIZE OF Y PARAMETER LABEL [8]
C-----AXES LABELS ARE PLACED IN THE MARGIN BETWEEN THE PLOT
C      AND WINDOW BOUNDARIES. THE CALLER SHOULD ENSURE THAT
C      THERE IS SUFFICIENT ROOM IN THESE MARGINS TO PLACE
C      SCALE LABELS AND PARAMETER LABELS ON THE LEFT (Y) AND
C      BOTTOM (X) OF THE PLOT.
C-----THE DEFAULT ORDINATE SETTINGS PROVIDE ROOM FOR 3 DIGITS
C      ON THE Y SCALE, A MINUS SIGN (IF NECESSARY), A SCALE
C      LABEL, AND THE ORDINATE PARAMETER LABEL. IF THE Y
C      VALUES ARE ALL POSITIVE, THE CALLER MAY SPECIFY 4 DIGITS
C      ON THE ORDINATE SCALE WITHOUT INCREASING THE DEFAULT
C      WINDOW BOUNDARIES.
C-----THE DEFAULT ABSCISSA SETTINGS PROVIDE ROOM FOR 3 DIGITS
C      ON THE X SCALE (NO ROOM FOR MINUS SIGN), THE X SCALE
C      LABEL, AND THE ABSCISSA PARAMETER LABEL. IF THE X
C      VALUES ARE NOT ALL POSITIVE, THE CALLER SHOULD DECREASE
C      THE NUMBER OF DIGITS ON THE ABSCISSA SCALE TO 2, OR
C      SPECIFY A LABEL FREQUENCY OF 2 OR MORE IN ORDER TO USE
C      THE DEFAULT WINDOW BOUNDARIES.
C      IP(27) INWP     MIN X ABS WINDOW BOUNDARY [85]
C      IP(28) IXWP     MAX X ABS WINDOW BOUNDARY [405]
C      IP(29) JNWP     MIN Y ABS WINDOW BOUNDARY [190]
C      IP(30) JXWP     MAX Y ABS WINDOW BOUNDARY [485]
C      IP(31) INPP     MIN X ABS PLOT BOUNDARY [140]
C      IP(32) IXPP     MAX X ABS PLOT BOUNDARY [395]
C      IP(33) JNPP     MIN Y ABS PLOT BOUNDARY [220]
C      IP(34) JXPP     MAX Y ABS PLOT BOUNDARY [475]
C*****PLOT TITLE
C-----AN N-LINE TITLE WINDOW, DEFAULTED FOR OUTPUT AT THE
C      BOTTOM OF THE PLOT, MAY BE ENABLED (DEFAULT) OR DISABLED
C      DEFAULT IS 3 TITLE LINES, EACH 45 CHAR LONG. NOTE THAT
C      BY CHANGING THE TITLE WINDOW BOUNDARIES, THE CALLER MAY
C      OUTPUT N LINES OF TEXT ANYWHERE ON THE OUTPUT PAGE.
C      IP(35) TTO      = 0  ENABLE TITLE OUTPUT []
C                      = 1  DISABLE TITLE OUTPUT
C      IP(36) TTOZ     SIZE OF TITLE OUTPUT [8]
C      IP(37) TTOL     LENGTH OF TITLE LINES [45]
C      IP(38) NTTO     NUMBER OF TITLE LINES [3]
C-----TITLE WINDOW
C      IP(39) INWT     MIN X ABS WINDOW BOUNDARY [85]
C      IP(40) IXWT     MAX X ABS WINDOW BOUNDARY [405]
C      IP(41) JNWT     MIN Y ABS WINDOW BOUNDARY [100]
C      IP(42) JXWT     MAX Y ABS WINDOW BOUNDARY [170]
C*****DISPOSITION OF THE PLOT FILE

```

```

C          IP(43)  SAVE   = 0   SAVE PLOT FILE []
C                               = 1   DISPOSE PLOT FILE
C          IP(44)  PRINT  = 0   DISABLE HARDCOPY []
C                               = 1   ENABLE HARDCOPY
C*****END PLOT SPECIFICATIONS
C
C          TX [C*1] ARRAY CONTAINING ALL TEXT TO BE OUTPUT DURING A RUN.
C          TEXT STRINGS ARE PLACED AT THE FOLLOWING OFFSETS:
C          RUNDATE BANNER : 0
C          X PARM LABEL   : RDOL
C          Y PARM LABEL   : RDOL + XPL
C          1ST TITLE LINE : RDOL + XPL + YPL
C          NTTO TTL LINE  : RDOL + XPL + YPL + (TTOL * (NTTO - 1))
C          LENGTH OF TX = RDOL + XPL + YPL + (TTOL * NTTO)
C          ** NOTE: TEXT WAS PLACED IN A SINGLE ARRAY TO ENABLE THE
C                   ARGUMENT LIST TO FIT ON A SINGLE LINE FOR S-MERGE.
C                   THE CALLING PROGRAM MAY USE THE EQUIVALENCE STATE-
C                   MENT TO ADDRESS INDIVIDUAL STRINGS. E.G., FOR A
C                   MAINLINE USING THE DEFAULT SETTING :
C                   CHARACTER *1  XPML(30),YPML(30),TTL(45,3),TX(195)
C                   EQUIVALENCE   (TX(1),XMPL(1)),(TX(31),YPML(1)),
C                   +              (TX(61),TTL(1,1))
C EXIT
C          ENTRY ARGUMENTS UNCHANGED.
C SUBPROGRAM REFERENCES
CALLS INITT      (4)
CALLS TWINDO     (4)
CALLS XYPDAT     (0)
CALLS MOVABS     (4)
CALLS MESSAGE    (3)
CALLS SCAXLR     (1)
CALLS SCLWIN     (1)
CALLS DWINDO     (4)
CALLS DRAXLR     (1)
CALLS PLOTXY     (1)
CALLS PLOTPT     (1)
CALLS LBAXLR     (1)
CALLS NCODI1     (2)
CALLS PFINIS     (4)
CALLS HDCOPY     (4)
CALLS DXNUM      (2)
CALLS DELETE     (2)
C-----
SUBROUTINE XYLOT (N,X,Y,VM,XN,XX,YN,YX,OPF,PF,IP,TX)
INTEGER          N, IP(44)
REAL             X(1),Y(1),VM,XN,XX,YN,YX
CHARACTER        OPF*14,PF*14,TX(1)*1,SC*24,CS(11)*1,XS(24)*1,
-              YS(24)*1,KEEP(2)*4,PFNAM*14,CHR*1
C
INTEGER          IPD(44),IPP(44),STAT,DRIVE,STATUS,DXNUM

```

```

INTEGER      RDO,RDOZ,RDOL,INWD,IXWD,JNWD,JXWD
INTEGER      LPO,LT,PPO,PPC,PPCZ,AXES,TICK
INTEGER      NDX,DXZ,NTX,FTX,XPL,XPLZ
INTEGER      NDY,DYZ,NTY,FTY,YPL,YPLZ
INTEGER      INWP,IXWP,JNWP,JXWP,INPP,IXPP,JNPP,JXPP
INTEGER      TTO,TTOZ,TTOL,NTTO,INWT,IXWT,JNWT,JXWT
INTEGER      SAVE,PRINT

```

C

```

EQUIVALENCE (IPP(1),RDO),(IPP(2),RDOZ),(IPP(3),RDOL)
EQUIVALENCE (IPP(4),INWD),(IPP(5),IXWD),(IPP(6),JNWD)
EQUIVALENCE (IPP(7),JXWD)
EQUIVALENCE (IPP(8),LPO),(IPP(9),LT),(IPP(10),PPO)
EQUIVALENCE (IPP(11),PPC),(IPP(12),PPCZ)
EQUIVALENCE (IPP(13),AXES),(IPP(14),TICK)
EQUIVALENCE (IPP(15),NDX),(IPP(16),DXZ),(IPP(17),NTX)
EQUIVALENCE (IPP(18),FTX),(IPP(19),XPL),(IPP(20),XPLZ)
EQUIVALENCE (IPP(21),NDY),(IPP(22),DYZ),(IPP(23),NTY)
EQUIVALENCE (IPP(24),FTY),(IPP(25),YPL),(IPP(26),YPLZ)
EQUIVALENCE (IPP(27),INWP),(IPP(28),IXWP),(IPP(29),JNWP)
EQUIVALENCE (IPP(30),JXWP),(IPP(31),INPP),(IPP(32),IXPP)
EQUIVALENCE (IPP(33),JNPP),(IPP(34),JXPP)
EQUIVALENCE (IPP(35),TTO),(IPP(36),TTOZ),(IPP(37),TTOL)
EQUIVALENCE (IPP(38),NTTO)
EQUIVALENCE (IPP(39),INWT),(IPP(40),IXWT)
EQUIVALENCE (IPP(41),JNWT),(IPP(42),JXWT)
EQUIVALENCE (IPP(43),SAVE),(IPP(44),PRINT)

```

C

```

DATA IPD /0,8,0,85,405,500,510,0,0,0,43,10,0,4,2,8,10,1,30,8,
-        3,8,10,1,30,8,85,405,190,485,140,395,220,475,
-        0,8,45,3,85,405,100,170,0,0/
DATA NPP /44/
DATA SC /'Scale Factor = 10 ** '/
DATA KEEP /'KEEP','DISP'/

```

C

C

*BEGIN XYPLOT

C

C*****INITIALIZE PLOT LIBRARY*****

1000 CALL INITT (OPF,PF)

C

C*****INITIALIZE PLOTTING PARAMETERS*****

```

DO 1010 I = 1, NPP
  IF (IP(I) .EQ. 0) THEN
    IPP(I) = IPD(I)
  ELSE IF (IP(I) .GT. 0) THEN
    IPP(I) = IP(I)
  ELSE IF (IP(I) .LT. 0) THEN
    IPP(I) = 0
  ENDIF

```

1010 CONTINUE

```

IR = IXPP - INPP
JR = JXPP - JNPP

C
C*****BUILD RUNDATE BANNER*****
2000 IF (RDO .EQ. 0) THEN
      CALL TWINDO (INWD,IXWD,JNWD,JXWD)
      CALL XYPDAT (INWD,JNWD,RDOZ)
    ELSE IF (RDO .EQ. 2) THEN
      CALL MOVABS (INWD,JNWD)
      CALL MESSAGE (INWD,JNWD,TX(1),RDOL,RDOZ,0.0,1)
    ENDIF

C
C*****SET DATA WINDOWS*****
C-----SET UP ABSCISSA SCALE
3000 XNPP = XN
      XXPP = XX
      CALL SCAXLR (N,X,VM,XNPP,XXPP,NTX,NDX,XSS,INS,ISF)
C-----SET UP ORDINATE SCALE
      YNPP = YN
      YXPP = YX
      CALL SCAXLR (N,Y,VM,YNPP,YXPP,NTY,NDY,YSS,JNS,JSF)
C-----SET PLOT WINDOW BOUNDARIES
      CALL SCLWIN (INPP,IXPP,INWP,IXWP,XNPP,XXPP,XNWP,XXWP)
      CALL SCLWIN (JNPP,JXPP,JNWP,JXWP,YNPP,YXPP,YNWP,YXWP)
      CALL TWINDO (INWP,IXWP,JNWP,JXWP)
      CALL DWINDO (XNWP,XXWP,YNWP,YXWP)

C
C*****BUILD AND LABEL THE AXES*****
C-----DRAW THE AXES
      IF (AXES .EQ. 0) THEN
        CALL DRAXLR (YNPP,XNPP,XSS,INS,TICK,1)
        CALL DRAXLR (YXPP,XNPP,XSS,INS,TICK,1)
        CALL DRAXLR (XNPP,YNPP,YSS,JNS,TICK,2)
        CALL DRAXLR (XXPP,YNPP,YSS,JNS,TICK,2)
C-----LABEL THE AXES
        CALL LBAXLR (XNPP,XSS,YNPP,0.0,INS,ISF,NDX,DXZ,TICK,FTX,1)
        CALL LBAXLR (XNPP,0.0,YNPP,YSS,JNS,JSF,NDY,DYZ,TICK,FTY,1)
C-----BUILD X SCALE FACTOR
        READ(SC,'(24A1)') (XS(J),J=1,24)
        CALL NCOD11 (ISF,CS)
        J = 9
        CS(J) = CS(1)
        DO 3100 I = 22, 24
          XS(I) = CS(J)
          J = J + 1
3100 CONTINUE
        IX = INPP
        IY = JNWP + XPLZ + 2
        IF ( ( (IR * 8) / (DXZ * 6) ) .GT. 24) THEN
          IX = IX + ( ( (IR * 8) / (DXZ * 6) - 24) / 2) * DXZ * 6 / 8

```

```

      ENDIF
      CALL MOVABS (IX,IY)
      CALL MESSAGE (IX,IY,XS,24,DXZ,0.0,1)
C-----BUILD Y SCALE FACTOR
      READ(SC,'(24A1)') (YS(J),J=1,24)
      CALL NCOD11 (JSF,CS)
      J = 9
      CS(J) = CS(1)
      DO 3200 I = 22, 24
        YS(I) = CS(J)
        J = J + 1
3200    CONTINUE
      IX = INWP + DYZ + YPLZ + 2
      IY = JNPP
      IF ( ( (JR * 8) / (DYZ * 6) ) .GT. 24) THEN
        IY = IY + ( ( (JR * 8) / (DYZ * 6) - 24) / 2) * DXZ * 6 / 8
      ENDIF
      CALL MOVABS (IX,IY)
      CALL MESSAGE (IX,IY,YS,24,DYZ,90.0,1)
    ENDIF
  C
  C*****BUILD THE PARAMETER LABELS*****
  C-----BUILD X PARAMETER LABEL
      IF (XPL .NE. 0) THEN
        IX = INPP
        IY = JNWP
        IF ( ( (IR * 8) / (XPLZ * 6) ) .GT. XPL) THEN
          IX = IX + (((IR*8) / (XPLZ*6) - XPL) / 2) * XPLZ * 6/8
        ENDIF
        CALL MOVABS (IX,IY)
        CALL MESSAGE (IX,IY,IX(RDOL+1),XPL,XPLZ,0.0,1)
      ENDIF
  C-----BUILD Y PARAMETER LABEL
      IF (YPL .NE. 0) THEN
        IX = INWP + YPLZ
        IY = JNPP
        IF ( ( (JR * 8) / (YPLZ * 6) ) .GT. YPL) THEN
          IY = IY + (((JR*8) / (YPLZ*6) - YPL) / 2) * YPLZ * 6/8
        ENDIF
        CALL MOVABS (IX,IY)
        CALL MESSAGE (IX,IY,IX(RDOL+XPL+1),YPL,YPLZ,90.0,1)
      ENDIF
  C
  C*****PLOT THE DATA*****
      IF (LPO .EQ. 0) CALL PLOTXY(N,X,Y,VM,LT)
      IF (PPO .EQ. 0) CALL PLOTPT(N,X,Y,VM,PPC,PPCZ)
  C
  C*****BUILD TITLE WINDOW*****
4000  IF (TTO .EQ. 0) THEN
      CALL TWINDO (INWT,IXWT,JNWT,JXWT)

```

```

IX = INWT
IY = JXWT
IO = RDOL + XPL + YPL - TTOL + 1
DO 4100 I = 1, NTTO
    IO = IO + TTOL
    IY = IY - (TTOZ + 2)
    CALL MOVABS (IX,IY)
    CALL MESSAGE (IX,IY,TX(IO),TTOL,TTOZ,0.0,1)
4100 CONTINUE
ENDIF
C
C*****OUTPUT THE PLOT FILE*****
IF (SAVE .NE. 0) STAT = 2
IF (SAVE .EQ. 0) STAT = 1
IF (PRINT .EQ. 0) CALL PFINIS(KEEP(STAT))
IF (PRINT .NE. 0) CALL HDCOPY(KEEP(STAT))
C-----STATUS='DISPOSE' DOES NOT WORK IN CURRENT FORTRAN
C          SO, WE'LL HAVE TO DELETE THE FILE OURSELVES.
IF (STAT .EQ. 2) THEN
    READ(PF,'(1X,A1)') CHR
    IF (CHR .EQ. ':') THEN
        READ(PF,'(A1)') CHR
        DRIVE = DXNUM(CHR)
        READ(PF,'(2X,A12)') PFNAM
    ELSE
        DRIVE = 0
        READ(PF,'(A12)') PFNAM
    ENDIF
    CALL DELETE(PFNAM,DRIVE,STATUS)
    IF (STATUS .EQ. 0) THEN
        WRITE(*,'(A,A14)') ' DELETE: FOLLOWING FILE DELETED : ',PF
    ELSE
        WRITE(*,'(A,A14)') ' DELETE REQUEST DENIED. FILE : ',PF
    ENDIF
ENDIF
C
C*TERMINATE XYPLOT
C
C    RETURN
C    END
C-----
C*****SUBROUTINE XYPDAT (INW,JNW,ISZ)
C-----
C 16 JUNE 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
C    XYPDAT BUILDS THE XYPLOT DEFAULT RUNDATE BANNER.
C ENTRY
C    INW [I]  ABSOLUTE MINIMUM X WINDOW BOUNDARY
C    JNW [I]  ABSOLUTE MINIMUM Y WINDOW BOUNDARY
C    ISZ [I]  SIZE OF RUNDATE BANNER
C EXIT

```

C ENTRY ARGUMENTS UNCHANGED

C SUBPROGRAM REFERENCES:

CALLS GDATE (5)

CALLS NCOD11 (2)

CALLS MOVABS (4)

CALLS MESSAGE (3)

```
C-----
      SUBROUTINE XYPDAT(INW,JNW,ISZ)
      CHARACTER          K*45,CM(12)*3,C(45)*1,CV(11)*1
      DATA CM /'JAN','FEB','MAR','APR','MAY','JUN',
-             'JUL','AUG','SEP','OCT','NOV','DEC'/
```

C

*BEGIN XYPDAT

C

```
      K = 'XYPLOT(840616)          RUN DATE : '
      READ(K,'(45A1)') (C(J),J=1,45)
      CALL GDATE (IYR,IMON,IDAY)
      CALL NCOD11 (IDAY,CV)
      C(35) = CV(10)
      C(36) = CV(11)
      READ(CM(IMON),'(3A1)') (C(J),J=38,40)
      CALL NCOD11 (IYR,CV)
      C(42) = CV(8)
      C(43) = CV(9)
      C(44) = CV(10)
      C(45) = CV(11)
      CALL MOVABS (INW,JNW)
      CALL MESSAGE (INW,JNW,C,45,ISZ,0.0,1)
```

C

*TERMINATE XYPDAT

C

```
C RETURN
END
```

Miscellaneous Plotting Routines

```
C-----
C*****SUBROUTINE DRAXLR (CF,CN,CS,NS,IT,ID)
```

C-----

C 10 October 1983 Orville D. Green IBM/PC DOS 2.0

C Description:

C DRAXLR draws a horizontal or vertical axis with linear spacing
C between tick marks. Beam position on return is at the origin.

C Arguments:

C CF	[real]	: fixed coordinate
C CN	[real]	: minimum coordinate of axis
C CS	[real]	: axis step size (distance between ticks)

```

C  NS   [integer]      : number of axis steps
C  IT   [integer]      : tick mark size (EPSON plot units)
C  ID   [integer]      : axis direction -
C                               ID = 1 : horizontal
C                               ID = 2 : vertical

```

C Subprogram references:

```

CALLS MOVEA   (3)
CALLS DRAWTA  (0)
CALLS DRAWA   (3)

```

C-----

```

      SUBROUTINE DRAXLR (CF,CN,CS,NS,IT,ID)
$DEBUG

```

*Begin DRAXLR

```

C      Initialize for horizontal draw :
      IF (ID .EQ. 1) THEN
          X0 = CN
          Y0 = CF
          X  = CN
          Y  = CF
          XS = CS
          YS = 0.0
          JD = 2

C      Initialize for vertical draw :
      ELSE IF (ID .EQ. 2) THEN
          X0 = CF
          Y0 = CN
          X  = CF
          Y  = CN
          XS = 0.0
          YS = CS
          JD = 1

C      Invalid parameter value passed. No draw.
      ELSE
          RETURN
      ENDIF

C      Draw the axis
      CALL MOVEA (X,Y)
      DO 10 I = 1, NS
          X = X + XS
          Y = Y + YS
          CALL DRAWA (X,Y)
          CALL DRAWTA (JD,IT,X,Y)
10      CONTINUE

```

CALL MOVEA (X0,Y0)

*Terminate DRAXLR

RETURN
END

C-----
C*****SUBROUTINE DRAWTA (ID,IT,X,Y)
C-----

C 10 October 1983 Orville D. Green IBM/PC DOS 2.0

C Description :

C DRAWTA draws horizontal or vertical tick marks using scaled
C coordinates. Beam position on return is at tick mark center.

C Arguments:

C ID [integer] : tick mark direction -
C ID = 1 : horizontal
C ID = 2 : vertical
C IT [integer] : tick mark size [EPSON plot units]
C X,Y [real] : position of tick mark center

C Subprogram references:

CALLS MOVEA (3)
CALLS BPOSN (0)
CALLS DRWTAB (0)
CALLS MOVABS (3)
CALLS DRWABS (3)

C-----

SUBROUTINE DRAWTA (ID,IT,X,Y)

\$DEBUG

*Begin DRAWTA

CALL MOVEA (X,Y)
CALL BPOSN (IX,IY)
CALL DRWTAB (ID,IT,IX,IY)

*Terminate DRAWTA

RETURN
END

C-----
C*****SUBROUTINE DRWTAB (ID,IT,IX,IY)
C-----

C 10 October 1983 Orville D. Green IBM/PC DOS 2.0

C Description :

C DRWTAB draws horizontal or vertical "tick" marks, using the

```

C      absolute coordinate scale. Beam position on return is at tick
C      mark center.

C Arguments :
C  ID      [integer]          : tick mark direction -
C                               ID = 1 : horizontal
C                               ID = 2 : vertical
C  IT      [integer]          : tick mark size (EPSON plot units)
C  IX,IY   [integer]          : absolute coordinates of tick mark center.
C-----

```

```

      SUBROUTINE DRWTAB (ID,IT,IX,IY)
$DEBUG

```

```

*Begin DRWTAB

```

```

C      set right,left,top,bottom coordinates to tick mark center
      IR = IX
      IL = IX
      JT = IY
      JB = IY

```

```

C      adjust right,left coordinates for horizontal tick
      IF (ID .EQ. 1) THEN
        IR = IR + IT/2
        IL = IR - IT

```

```

C      adjust top,bottom coordinates for vertical tick
      ELSE IF (ID .EQ. 2) THEN
        JT = JT + IT/2
        JB = JB - IT
      ENDIF

```

```

C      draw the tick mark
      CALL MOVABS (IR,JT)
      CALL DRWABS (IL,JB)
      CALL MOVABS (IX,IY)

```

```

*Terminate DRWTAB

```

```

      RETURN
      END

```

```

C-----
C*****SUBROUTINE LBAXLR (XN,XS,YN,YS,NS,ISF,NDGT,ISL,ITM,IFL,IRP)
C-----

```

```

C 10 October 1983  Orville D. Green  IBM/PC  DOS 2.0

```

```

C Description :

```

```

C  LBAXLR labels a linear horizontal or vertical axis.

```

```

C Arguments :
C XN [real]           : minimum x-coordinate of axis
C XS [real]           : x-axis step size (zero for vertical axis)
C YN [real]           : minimum y-coordinate of axis
C YS [real]           : y-axis step size (zero for horizontal axis)
C NS [integer]        : number of steps on axis
C ISF [integer]       : axis scaling factor
C NDGT [integer]      : number of digits in scale label
C ISL [integer]       : size of scale label (EPSON plot units)
C ITM [integer]       : size of tick marks (EPSON plot units)
C IFL [integer]       : frequency of axis labels
C IRP [integer]       : relative position -
C                     IRP = 1 : below/left of axis
C                     IRP = 2 : above/right of axis

```

C Subprogram references :

```

CALLS SCLVAL (0)
CALLS NCODI1 (1)
CALLS MOVEA (3)
CALLS BPOSN (0)
CALLS ABORT (3)
CALLS MOVABS (3)
CALLS MESSAGE (2)

```

C-----

```

SUBROUTINE LBAXLR (XN,XS,YN,YS,NS,ISF,NDGT,ISL,ITM,IFL,IRP)
$DEBUG

```

```

C declarations
CHARACTER CV(11) *1, CA *60

```

*Begin LBAXLR

```

C start at min, work to max at indicated frequency.

```

```

X = XN
Y = YN
DO 10 I = 1, NS + 1, IFL

```

```

C encode scale label
IF (XS .EQ. 0.0) V = Y
IF (YS .EQ. 0.0) V = X
CALL SCLVAL (V,NDGT,ISF,1,IV)
CALL NCODI1 (IV,CV)
LC = 11 - NDGT
CV(LC) = CV(1)

```

```

C set up coordinates for label output
CALL MOVEA(X,Y)
CALL BPOSN (IX,IY)

```

```

C      vertical axis:
      IF (XS .EQ. 0.0) THEN
        IY = IY - (ISL/2)
        IX = IX - (IRP * (NDGT+1) * ISL) - (IRP * ITM) / 2

C      horizontal axis:
      ELSE IF (YS .EQ. 0.0) THEN
        IX = IX - ( (NDGT + 1) * ISL) / 2
        IY = IY - (IRP * ISL) - (IRP * ITM) / 2

C      invalid arguments passed. abort plot.
      ELSE
        CA = '** LBAXLR ** Arguments XS and YS both non-zero'
        CALL ABORT (CA,60)
      ENDIF

C      Now write the label out, and increment for next position
      CALL MOVABS(IX,IY)
      CALL MESSAGE (IX,IY,CV(LC),NDGT+1,ISL,0.0,1)
      X = X + XS * REAL(IFL)
      Y = Y + YS * REAL(IFL)
10    CONTINUE

```

*Terminate LBAXLR

```

      RETURN
      END

```

```

C-----
C*****SUBROUTINE PLOTXY (N,X,Y,YM,LT)
C-----

```

C 10 October 1983 Orville D. Green IBM/PC DOS 2.0

C Description :

C PLOTXY plots a set of ordered (x,y) pairs.

C Arguments :

C N	[integer]	: number of (x,y) pairs
C X	[real]	: array containing abscissa values
C Y	[real]	: array containing ordinate values
C YM	[real]	: value assigned to a missing datum
C LT	[integer]	: DASHA dash type specification

C Subprogram references:

CALLS MOVEA (3)

CALLS DASHA (3)

```

C-----

```

```

      SUBROUTINE PLOTXY (N,X,Y,YM,LT)
$DEBUG

```

```
C      declarations
      REAL          X(1),Y(1)
```

```
*Begin PLOTXY
```

```
C      find first valid datum
      DO 10 I = 1, N
        IF (Y(I) .NE. YM) GOTO 20
10    CONTINUE
```

```
C      no valid data - return
      RETURN
```

```
C      plot the first valid datum
20    CALL MOVEA (X(I),Y(I))
      CALL DASHA (X(I),Y(I),LT)
```

```
C      plot subsequent data
      DO 30 J = I + 1, N
        IF (Y(J - 1) .EQ. YM) CALL MOVEA (X(J),Y(J))
        IF (Y(J)      .NE. YM) CALL DASHA (X(J),Y(J),LT)
30    CONTINUE
```

```
*Terminate PLOTXY
```

```
      RETURN
      END
```

```
C-----
C*****SUBROUTINE PLOTPT (N,X,Y,M,PC,PZ)
```

```
C-----
C 15 MAY 1984  ORVILLE D. GREEN  IBM/PC  DOS 2.0
```

```
C      PLOTPT PLOTS A "SCATTER DIAGRAM" OF ORDERED (X,Y) PAIRS.
```

```
C ENTRY
```

```
C      N [I]  NUMBER OF (X,Y) PAIRS
```

```
C      X [R]  ABSCISSA VALUE ARRAY
```

```
C      Y [R]  ORDINATE VALUE ARRAY
```

```
C      M [R]  VALUE ASSIGNED TO A MISSING DATUM
```

```
C      PC [I] PLOT CHARACTER (ASCII)
```

```
C      PZ [I] PLOT CHARACTER SIZE (EPSON PLOT UNITS)
```

```
C EXIT
```

```
C      ENTRY ARGUMENTS UNCHANGED
```

```
C SUBPROGRAM REFERENCES
```

```
CALLS MOVEA      (3)
```

```
CALLS BPOSN      (0)
```

```
CALLS MOVABS      (3)
```

```
CALLS MESSAGE     (2)
```

```
C-----
      SUBROUTINE PLOTPT (N,X,Y,M,PC,PZ)
```

```
$DEBUG
```

```
      INTEGER          N,PC,PZ
```

```

REAL          X(N),Y(N),M
CHARACTER     C*1

C
*BEGIN PLOTPT
C
  C = CHAR(PC)
  DO 10 I = 1, N
    IF ( (X(I) .NE. M) .AND. (Y(I) .NE. M) ) THEN
      CALL MOVEA (X(I),Y(I))
      CALL BPOSN (IX,IY)
      IX = IX + PZ/6
      CALL MOVABS (IX,IY)
      CALL MESSAGE (IX,IY,C,1,PZ,0.0,5)
    ENDIF
  10 CONTINUE
C
*TERMINATE PLOTPT
C
  RETURN
  END

-----
C*****SUBROUTINE BPOSN (KX,KY)
-----
C 10 October 1983  Orville D. Green  IBM/PC  DOS 2.0
C Update

C Description :
C   BPOSN addresses the PLIBMEM common block to get the current
C   beam position.

C Arguments :
C   KX [integer]      : absolute x-coordinate of beam position
C   KY [integer]      : absolute y-coordinate of beam position
-----
C   SUBROUTINE BPOSN (KX,KY)
$DEBUG

C   declarations
  INTEGER          CREC,UNIT
  COMMON          /PLTDTA/ LX,LY,SX,SY,IN,IX,JN,JX,XN,XX,YN,YX,CREC,
1                  UNIT,ANGL,ANGS,ANGC

*Begin BPOSN

  KX = LX
  KY = LY

*Terminate BPOSN

  RETURN

```

END

```

C-----
C*****SUBROUTINE SCAXLR(N,D,DM,SN,SX,NST,NDGT,SS,NS,ISF)
C-----
C 10 October 1983  Orville D. Green  IBM/PC  DOS 2.0

C Description :
C   SCAXLR builds a linear scale. The scale depends on :
C   (i)  the number of digits to appear in the scale label,
C   (ii) the number of steps desired on the axis.
C   (iii) the step value.
C   The scale is built such that the scale comes out "even", e.g.,
C       36 46 56   is set to 30 40 50 60
C       31 41 51   is set to 25 35 45 55

C Arguments:
C   N      [integer]      : size of data array
C   D      [real]         : array of data values
C   DM     [real]         : value assigned to a missing datum
C   SN     [real]         : desired scale minimum
C   SX     [real]         : desired scale maximum
C                                     *note: SN,SX may be adjusted to even scale out
C   NST    [integer]      : target number of steps
C   NDGT   [integer]      : number of digits in scale label
C   SS     [real]         : scale step size
C   NS     [integer]      : actual number of steps
C   ISF    [integer]      : scaling factor

C Subprogram references:
CALLS RMXMN1      (1)
CALLS SCLZRD      (0)
CALLS SCLNRD      (0)
C-----

```

```

      SUBROUTINE SCAXLR(N,D,DM,SN,SX,NST,NDGT,SS,NS,ISF)
$DEBUG

```

```

C   declarations:
      REAL      D(1)

```

*Begin SCAXLR

```

C   Find the min and max data values. If the actual data limits fall
C   outside the passed scale limits, then adjust the scale limits.
      CALL RMXMN1 (D,N,DM,DN,DX)
      IF (DX .NE. DM) THEN
         IF (DN .LT. SN) SN = DN
         IF (DX .GT. SX) SX = DX
      ENDIF

```

```

C      find the scale for the current scale limits.
100    DR = SX - SN
      IF (DR .EQ. 0.0) CALL SCLZRD(SN,SX,NST,NDGT,SS,NS,ISF)
      IF (DR .NE. 0.0) CALL SCLNRD(SN,SX,NST,NDGT,SS,NS,ISF)

```

```

*Terminate SCAXLR

```

```

      RETURN
      END

```

```

C-----
C*****SUBROUTINE SCLZRD (SN,SX,NST,NDGT,SS,NS,ISF)
C-----
C 10 October 1983  Orville D. Green  IBM/PC  DOS 2.0

```

```

C Description :

```

```

C  SCLZRD builds a scale for data having zero range. (see SCAXLR)

```

```

C Arguments :

```

```

C  SN      [real]           : minimum scale value
C  SX      [real]           : maximum scale value
C  NST     [integer]        : number of steps desired on scale
C  NDGT    [integer]        : number of digits in scale label
C  SS      [real]           : scale step size
C  NS      [integer]        : number of scale steps
C  ISF     [integer]        : scaling factor

```

```

C Subprogram references:

```

```

CALLS SCLVAL      (0)
CALLS RNDVAL      (1)
CALLS RP10        (1)
CALLS ROUND       (1)

```

```

C-----

```

```

      SUBROUTINE SCLZRD (SN;SX,NST,NDGT,SS,NS,ISF)
$DEBUG

```

```

*Begin SCLZRD

```

```

C      zero-valued data
      IF (SX .EQ. 0.0) THEN
        NS = NST
        SS = 1.0
        SN = 0.0 - REAL(NS/2) * SS
        SX = SN + REAL(NS) * SS
        ISF = 0
        GOTO 99999

```

```

C      non-zero valued data
      ELSE

```

```

C      positive-valued data
100    IF (SX .GT. 0.0) THEN
        SN = 0.0
        CALL SCLVAL(SX,NDGT,ISF,-1,IX)
        ST = RP10(REAL(IX),ISF)
        IF (SX .NE. ST) THEN
            SX = ST
        ENDIF

C      negative-valued data
      ELSE
        SX = 0.0
        CALL SCLVAL(SN,NDGT,ISF,-1,IN)
        ST = RP10(REAL(IN),ISF)
        IF (SN .NE. ST) THEN
            SN = ST
        ENDIF
      ENDIF

C      find scal step size and number of steps
      SS = ST / REAL(NST)
      SS = RNDVAL(SS,0.5,0)
      NS = INT(ROUND(ST/SS,0,0.1))

C      even the scale out, if necessary
      IF (AMOD(ST,SS) .NE. 0.0) THEN
        NS = NS + 1
        IF (SX .GT. 0.0) THEN
            SX = SS * REAL(NS)
            ST = SX
        ELSE
            SN = 0.0 - SS * REAL(NS)
            ST = SN
        ENDIF
        CALL SCLVAL(ST,NDGT,ISFT,-1,IT)
        IF (ISFT .NE. ISF) GOTO 100
      ENDIF
    ENDIF
  ENDIF

```

*Terminate SCLZRD

```

99999 CONTINUE
      RETURN
      END

```

```

C-----
C*****SUBROUTINE SCLNRD (SN,SX,NST,NDGT,SS,NS,ISF)
C-----
C 10 October 1983  Orville D. Green  IBM/PC  DOS 2.0

```

C Description:

C SCLNRD builds a scale for data having non-zero range. (see SCAXLR)

C Arguments:

C SN [real] : minimum scale value
 C SX [real] : maximum scale value
 C NST [integer] : target number of scale steps
 C NDGT [integer] : number of digits in the scale label
 C SS [real] : scale step size
 C NS [integer] : actual number of scale steps
 C ISF [integer] : scaling factor

C Subprogram references:

CALLS SCLVAL (0)
 CALLS RNDVAL (1)
 CALLS ROUND (1)
 CALLS RP10 (1)

C-----
 SUBROUTINE SCLNRD (SN,SX,NST,NDGT,SS,NS,ISF)
 \$DEBUG

*Begin SCLNRD

C find the scaling factor
 100 IF (SX .NE. 0.0) THEN
 CALL SCLVAL(SX,NDGT,ISF,-1,IX)
 CALL SCLVAL(SN,NDGT,ISF, 1,IN)
 ELSE
 CALL SCLVAL(SN,NDGT,ISF,-1,IN)
 CALL SCLVAL(SX,NDGT,ISF, 1,IX)
 ENDIF

 C adjust the maximum and minimum values, if necessary
 SXT = RP10(REAL(IX),ISF)
 SNT = RP10(REAL(IN),ISF)
 IF (SX .NE. SXT) THEN
 SX = SXT
 ENDIF
 IF (SN .NE. SNT) THEN
 SN = SNT
 ENDIF

 C find the step size and number of steps
 SR = SX - SN
 SS = SR / REAL(NST)
 SS = RNDVAL(SS,0.5,0)
 NS = INT(ROUND(SR/SS,0,0.1))

 C even the scale out, if necessary
 IF (SX .EQ. 0.0) THEN
 IF (AMOD(SN,SS) .NE. 0.0) THEN
 NS = NS + 1

```

      SN = 0.0 - REAL(NS) * SS
    ENDIF
  ELSE IF (SN .EQ. 0.0) THEN
    IF (AMOD(SX,SS) .NE. 0.0) THEN
      NS = NS + 1
      SX = REAL(NS) * SS
    ENDIF
  ELSE IF ( (SX .GT. 0.0) .AND. (SN .LT. 0.0) ) THEN
    NSX = INT(ROUND(SX/SS,0,0.1))
    NSN = INT(ROUND(SN/SS,0,0.1))
    NSX = IABS(NSX)
    NSN = IABS(NSN)
    IF (AMOD(SX,SS) .NE. 0.0) THEN
      NSX = NSX + 1
      SX = REAL(NSX) * SS
    ENDIF
    IF (AMOD(SN,SS) .NE. 0.0) THEN
      NSN = NSN + 1
      SN = 0.0 - REAL(NSN) * SS
    ENDIF
    NS = NSX + NSN
  ELSE
    IF (AMOD(SX-SN,SS) .NE. 0.0) THEN
      NS = NS + 1
      SX = SN + REAL(NS) * SS
    ENDIF
  ENDIF
  IF (SX .NE. 0.0) THEN
    CALL SCLVAL(SX,NDGT,ISFT,-1,IX)
  ELSE
    CALL SCLVAL(SN,NDGT,ISFT,-1,IN)
  ENDIF
  IF (ISFT .NE. ISF) GOTO 100

```

*Terminate SCLNRD

```

  RETURN
END

```

```

C-----
C*****SUBROUTINE SCLVAL(V,N,ISF,IC,IV)
C-----
C 10 October 1983  Orville D. Green  IBM/PC  DOS 2.0

```

C Arguments:

```

C  V      [real]          : value to scale
C  N      [integer]       : number of digits in scale
C  ISF    [integer]       : scaling factor (passed or returned : f(IC))
C  IC     [integer]       : processing code:
C                          IC ¼ 0      : determine scaling factor
C                          IC ½ 0      : use passed scale factor

```

```

C          ?IC? = 1 : round V up
C          ?IC? = 2 : round V down
C          ?IC? = 3 : normal V round

```

C Subprogram references:

```

CALLS ROUND (1)
CALLS IMAG (1)
CALLS RP10 (1)

```

```

C-----
SUBROUTINE SCLVAL(V,N,ISF,IC,IV)
$DEBUG

```

```

C  declarations:
REAL F(3)
DATA F /0.9,0.1,0.5/

```

*Begin SCLVAL

```

IF (V .EQ. 0.0) THEN
  IV = 0
  IF (IC .LT. 0) ISF = 0
ELSE
  J = IABS(IC)
  VA = ABS(V)
  IPV = IMAG(VA)
  VN = RP10(VA,-IPV)
  IF (IC .LT. 0) THEN
    IF (VN .GE. 1.0) NR = N - 1
    IF (VN .LT. 1.0) NR = N
    VR = ROUND(VN,NR,F(J))
    VRS = RP10(VR,NR)
    VNS = RP10(VN,NR)
    IPR = IMAG(VRS)
    IPN = IMAG(VNS)
    IF (IPR .GT. IPN) NR = NR - 1
    ISF = IPV - NR
  ELSE
    NR = IPV - ISF
    VR = ROUND(VN,NR,F(J))
  ENDIF
  IV = INT(RP10(VR,NR) * V/VA)
ENDIF

```

*Terminate SCLVAL

```

RETURN
END

```

```

C-----
C*****SUBROUTINE SCLWIN (IDNP,IDX P,IDNW,IDXW,DNP,DXP,DNW,DXW)
C-----

```

C 10 October 1983 Orville D. Green IBM/PC DOS 2.0

C Description:

C SCLWIN finds the scaled window boundaries given the scaled
C plot boundaries and absolute window boundaries. (Useful
C for putting margins on plots)

C Arguments:

C IDNP	[integer]	: minimum plot absolute coordinate
C IDXP	[integer]	: maximum plot absolute coordinate
C IDNW	[integer]	: minimum window absolute coordinate
C IDXW	[integer]	: maximum window absolute coordinate
C DNP	[real]	: minimum plot scaled coordinate
C DXP	[real]	: maximum plot scaled coordinate
C DNW	[real]	: minimum window scaled coordinate (return)
C DXW	[real]	: maximum window scaled coordinate (return)

C-----

SUBROUTINE SCLWIN (IDNP, IDXP, IDNW, IDXW, DNP, DXP, DNW, DXW)
\$DEBUG

*Begin SCLWIN

DPU = (DXP - DNP) / REAL(IDXP - IDNP)
DNW = DNP - (DPU * REAL(IDNP - IDNW))
DXW = DXP + (DPU * REAL(IDXW - IDXP))

*Terminate SCLWIN

RETURN
END

C-----

C*****REAL FUNCTION ROUND (V,I,F)

C-----

C 10 October 1983 Orville D. Green IBM/PC DOS 2.0

C Description:

C ROUND rounds a value to the indicated decimal position.

C Arguments:

C V	[real]	: value to round
C I	[integer]	: decimal position to round to -
C		$\frac{1}{2}0$: round to I places right of decimal point.
C		$\frac{1}{4}=0$: round to I+1 places left of decimal point.
C F	[real]	: rounding factor -
C		0.5 = "normal" round
C		0.9 = round "up"
C		0.0 = round "down"

C Subprogram references :

CALLS RP10 in dataset UTLTYLBR.FOR [1.00]

C-----

REAL FUNCTION ROUND (V,I,F)

*Begin ROUND

```

IF (V .NE. 0.0) THEN
  SIGN = V / ABS(V)
  T = REAL(INT(F+RP10(ABS(V),I)))
  ROUND = SIGN * RP10(T,-I)
ELSE
  ROUND = 0.0
ENDIF

```

*Terminate ROUND

```

RETURN
END

```

C-----

C*****REAL FUNCTION RP10 (V,I)

C-----

C 10 October 1983 Orville D. Green IBM/PC DOS 2.0

C Description:

C RP10 raises a real value to an integer power of 10. (allows
C negative exponents to be used).

C Arguments:

C V [real] : value to be raised
C I [integer] : power of 10 by which to raise V

C-----

REAL FUNCTION RP10 (V,I)

*Begin RP10

```

VT = V
IP = IABS(I)
DO 10 J = 1, IP
  IF (I .LT. 0) THEN
    VT = VT / 10.0
  ELSE IF (I .GT. 0) THEN
    VT = VT * 10.0
  ENDIF
10 CONTINUE
RP10 = VT

```

*Terminate RP10

RETURN
END

C-----
C*****REAL FUNCTION RNDVAL (V,S,I)
C-----
C 10 October 1983 Orville D. Green IBM/PC DOS 2.0

C Description:

C RNDVAL rounds a value to the indicated specifications

C Arguments:

C V [real] : value to round
C S [real] : rounding specification *
C I [integer] : 0= $\frac{1}{2}$ round up, 1= $\frac{1}{2}$ round down

C *S = 0.5, I = 0 : V = 123 rounded to 150
C : V = 1.2 rounded to 1.5

C Subprogram references:

CALLS IMAG in dataset UTLTYLBR.FOR [1.00]
CALLS RP10 in dataset UTLTYLBR.FOR [1.00]

C-----

REAL FUNCTION RNDVAL (V,S,I)

*Begin RNDVAL

```

      IF ( (S .LE. 0.0) .OR. (I .LT. 0) .OR. (I .GT. 1) ) VA = 0.0
      ISIGN = 1.0
      VA = ABS(V)
      IF (VA .NE. 0.0) THEN
        SIGN = V / VA
        IP10 = IMAG(VA)
        STEP = RP10(S,IP10)
        VT = 0.0
10      VT = VT + STEP
        IF ( (I .EQ. 0) .AND. (VT .LT. VA) ) GOTO 10
        IF ( (I .NE. 0) .AND. (VT .LE. VA) ) GOTO 10
        VA = VT
      ENDIF
      RNDVAL = VA * SIGN

```

*Terminate RNDVAL

RETURN
END

```

C-----
C*****INTEGER FUNCTION IMAG (V)
C-----
C Version 1.00 (10 October 1983)          by Orville D. Green
C Update

C Description:
C   IMAG returns the order of magnitude of a value.
C   (required because ALOG10(10.0) = 0.99999...)

C Arguments:
C   V [real]      : value

C-----

```

INTEGER FUNCTION IMAG (V)

*Begin IMAG

```

      VA  = ABS(V)
      IMAG = 0
      IF (VA .GE. 10.0) THEN
10      IMAG = IMAG + 1
         VA  = VA / 10.0
         IF (VA .GE. 10.0) THEN
            GOTO 10
         ELSE
            GOTO 99999
         ENDIF
      ELSE IF (VA .LT. 1.0) THEN
20      IMAG = IMAG - 1
         VA  = VA * 10.0
         IF (VA .LT. 1.0) THEN
            GOTO 20
         ELSE
            GOTO 99999
         ENDIF
      ENDIF

```

*Terminate IMAG

```

99999 RETURN
      END

```

A CASE-STUDY ANALYSIS OF SURFACE/UPPER-A

3 1762 00512724 4

Main
N378 Green, Orville D.
G825 A case-study analysis
cop.2 of surface /upper-air...

[illegible]

Main
N378
G825
cop.2