44913 19940018775

N94-23248

1993 NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

JOHN F. KENNEDY SPACE CENTER
UNIVERSITY OF CENTRAL FLORIDA

513-47
197264
P. 33

HURRICANE RISK ASSESSMENT: ROLLBACK OR RIDE OUT

PREPARED BY: Mr. Richard A. Wohlman

ACADEMIC RANK: Instructor

UNIVERSITY AND DEPARTMENT: Western Carolina University Department of Mathematics and Computer Science

NASA/KSC

DIVISION: Advanced Projects Office

BRANCH: Atmospheric Sciences

NASA COLLEAGUE: Jim Nicholson

DATE: August 8, 1993

CONTRACT NUMBER: University of Central Florida NASA-NGT-60002 Supplement: 11

ACKNOWLEDGEMENTS

Just a moment to thank some folks-

To the UCF people, Ray and Kari, pretty smooth operation you two.

To my NASA colleague Jim Nicholson. Even after the move to the other end of the world, I managed to get something done!

To Rich Hall for his knowledge and assistance in Turbo Pascal.

To all of the other Faculty Fellows for good fellowship and encouragement and tolateration!

To my fellow paraplane pilots, Martha, Greg and Keith, when are we gonna do this again?

And to my special friends and troublemakers Keith (yeah, I know he shows up twice) and Harry a truly sincere thanks for the boost to not only my ego but to fanning the fires of intellectual persuits.

This has been two of the most exciting and enjoyable summers of my life, and I congratulate all who have had the opportunity to participate and contribute. My best wishes go to each and every one of you. Thank you again.

RAW
8/8/93

ABSTRACT

Winds in excess of 74.5 knots could cause severe damage to a space shuttle on the launch pad. Current plans exist for rollback to the Vehicle Assembly Building, but require 48 hour leadtime to implement. Decisions based upon cost/loss are evaluated to ascertain whether predetermined forecast probabilities for rollback/rideout decisions can be made far in advance of hurricane season for use in decision making.

SUMMARY

KSC lies in the hurricane belt, and there exists the possibility that a hurricane with winds of at least 74.5 knots might strike the Center. Should a Space Shuttle be on the launch pad during winds of this nature, severe damage up to and including the loss of the vehicle could occur. Therefore, in the event that winds associated with a hurricane are forecast to be in excess of 74.5 knots, procedures are in place to roll the mobile launch platform with the shuttle on top back to the relative safety of the Vehicle Assembly Building.

This plan requires 48 notification in the advance of the onset of the winds, with the last eight of those hours being devoted to the actual movement of the shuttle from the pad to the VAB.

Managers make the rollback/rideout decision based upon National Hurricane Center forecasts, which give numerical percentages of strike. At present, there is no predetermined probability value which triggers the rollback decision, and at present the decision is extremely subjective.

With a Bayesian analysis of sequential events and their modification to current probabilities, along with a cost versus loss study, an objective value from which to base a decision might be reached. In addition, examination of past information concerning similar hurricane paths might be useful in fine tuning the probability forecasts

TABLE OF CONTENTS

SECTION I INTRODUCTION

1.1 BACKGROUND AND PROBLEM IDENTIFICATION

The Kennedy Space Center, located on the east coast of the Florida peninsula has been subjected to 64 tropical storms in years 1886 through 1992. Of these, 26 were of hurricane strength. Because winds in excess of 74 knots would cause extreme damage to a shuttle on the launch pad, protective measures in the form of rolling the shuttle back to the Vehicle Assembly Building are planned in the event of such winds. That rollback decision has to be made by shuttle managers in sufficient time to allow the actual rollback, and the current timeline (appendix 1) requires that decision be made 48 hours prior to the onset of damaging winds.

The primary responsibility of hurricane forecasting lies with the National Hurricane Center (NHC) who distributes forecasts via alerts, watches and warnings to the various meteorological stations around the country. Weather support to the Kennedy Space Center is provided directly by personnel at the Cape Canaveral Air Force Station Forecast Facility. These meteorologists provide the manager with not only forecasts, but rationale, track error analysis, and model confidence.

While hurricane forecasting has become quite sophisticated in recent years, there are still sufficient forecast positioning errors to preclude a cut and dried decision to rollback or to ride out. For example, a recent National Hurricane Center study shows RMS position errors to be about 170 nautical miles at the 24 hour point. As the radius of maximum winds in a 95 knot hurricane is less than 25 nautical miles, a storm of that nature moving inland at 70 miles to the north or south of the center would produce winds well below the maximum allowable. Thus, even though the forecast might place the storm directly atop KSC in 24 hours, managers cannot use that forecast as the sole source for decision making.

Because rolling the shuttle back to the Vehicle Assembly Building is such a costly event, both in terms of launch delays, shuttle processing schedules, and redundant operations, the rollback option is taken only after much deliberation. And, because hurricane strikes at the cape are relatively rare occurrences, managers have not been able to create any kind of track record of past decisions and the criterion upon which those decisions were based. Managers rely on advice provided by the weather support personnel which in turn is largely based on Hurricane Center forecasts. One problem arises with the probability forecast itself which is given as a probability of landfall at a certain area. Does the manager protect if the probability is 5%? What about 10%? At present, there is no black and white set of rules for making the rollback decision based strictly upon the strike probability.

The situation is further muddied by the aforementioned forecast position errors. Obviously, the farther into the future you forecast, the larger the position errors. This necessarily produces low strike probabilities for specific locations due to the uncertainties involved.

Simply put, the manager needs some method to yield a set of guidelines for use in making the rollback/rideout decision at the 48 hour point.

1.2 PREVIOUS STUDIES

Investigation into the work done in the area was conducted during the summer of 1992 here at the Kennedy Space Center. A detailed breakdown of several approaches is discussed in the report of that work. The bulk of the data was found in the NOAA Technical Memorandum NWS NHC 38 by Charles J. Neumann titled 'The National Hurricane Center Risk Analysis Program (HURISK)'.

HURISK is a computer program developed to find statistical data concerning, strangely enough, hurricane risk assessment. When conducted for a specific location, it provides data about the frequency, strength and movement of tropical storms through a circle 75 nautical miles in radius around the selected site. Of particular interest are the data computed for the mean return periods (chart 1) of tropical storms with winds of a specified amount, and the probability of at least *x* storms passing within 75 nautical miles of KSC over *n* consecutive years (chart 2). This second chart is important as it also yields the probability of going for *n* consecutive years without having a storm of a specified category. See the source document for detailed explanation.

In addition, computerized tropical storm track data were obtained which contained information similar to that found in the NOAA Historical Climatology Series 6-2, Tropical Cyclones of the North Atlantic Ocean, 1871-1986'. These data were examined in a rather cursory

SITE: CAPE KENNEDY, FL

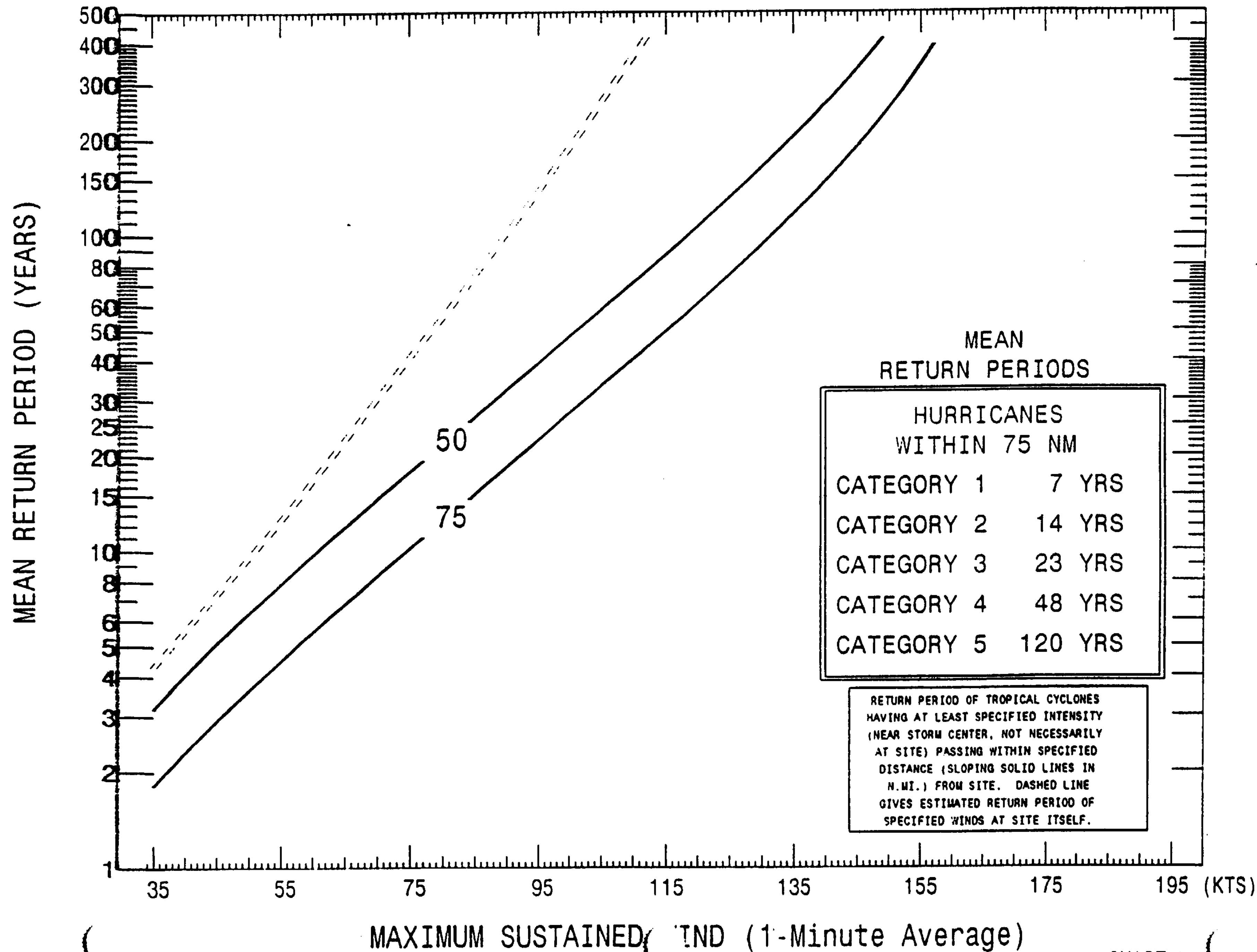


CHART I

PROBABILITY OF AT LEAST X TROPICAL CYCLONES (≧64 KNOTS) PASSING
WITHIN 75 N.MI. OF SPECIFIED SITE OVER N CONSECUTIVE YEARS

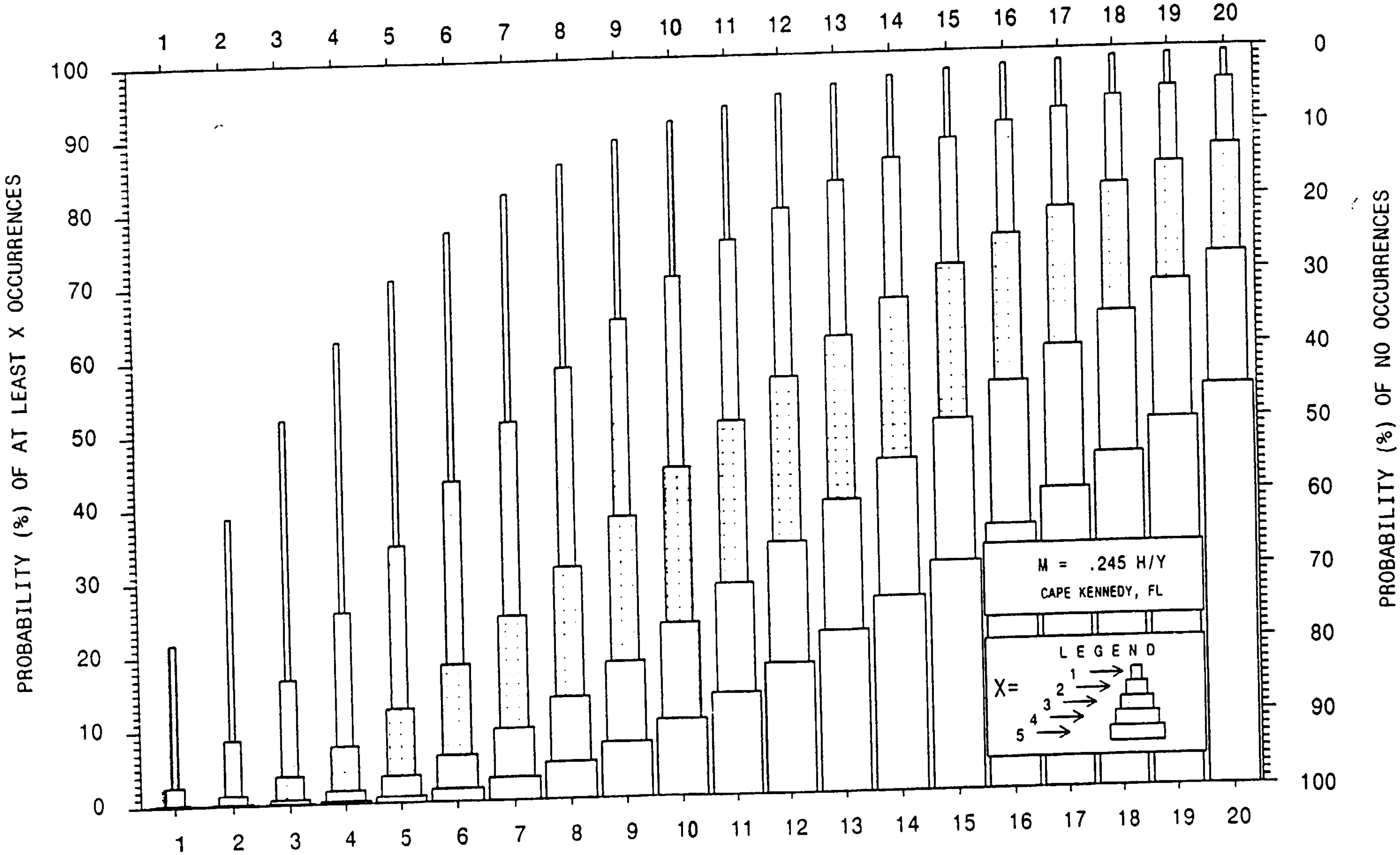


CHART 2

manner as rudimentary programs were produced to select specific storm data. It was concluded that these data should be more closely examined in a graphical format, and that by selecting storms similar in nature to the ongoing storm, it might be possible to detect trends from statistical fields generated from those similar storms.

1.3 NEW TOOLS FOR THE DECISION MAKER

One approach to giving the decision maker a go/no-go kind of decision is by using a cost versus loss analysis. Typically, this kind of analysis is used by managers in determining whether to initiate some kind of protection scheme strictly from a monetary point of view. This method determines the most cost effective course of action based solely on the dollars involved, and ignores the cost of public opinion, loss of capability, and the downstream effects of the loss itself. Additionally, by applying a Bayesian analysis to the probability of having a damaging hurricane throughout the season, new thresholds can be developed which yield probabilities for using the cost versus loss decision making process.

In the event that protection methods are initiated, costs can be considered to be delays in the shuttle flow process. In fact, these delays are not simply the time it takes to enact some protection to the shuttle, but must include the time that it takes to bring the shuttle to the same place in the processing timeline that it was in prior to initiating that protection. The very nature of the differences in payload requirements themselves makes it difficult to obtain dollar values for the protection operations. For example, a spacelab mission which is loaded horizontally in the Orbiter Processing Facility would not need to be removed from the orbiter while still on the pad. However, a Tracking and Data Relay Satellite, which is inserted vertically into the payload bay once the shuttle moves to the launch pad might have to be removed from the orbiter prior to rolling the orbiter back to the VAB. For the purposes of this study, cost will be assumed to $2 million per day[1].

In addition to the statistical study, a graphical package was under development to assist meteorologists in reviewing historical tropical storm data. The computer programs are designed to permit individual selection of storms based upon specified criterion, to allow the

---

1 Because the costs are so variable, perhaps a better currency for decision making is days instead of dollars.

forecaster to create statistical data on storms which show similarities to the current storm. Several interesting problems needed to be overcome, and work continues in this area. Once complete, these should provide the forecaster the capability to modify the forecast based upon movement, and a statistical study of forecast and observed motion vectors of the current storm. At any rate, graphical representation of historical data is accomplished, and should help in storm track visualization.

SECTION II DECISION MAKING

2.1 SIMPLE COST VERSUS LOSS

The cost versus loss approach has been applied to decision making with respect to meteorological events for some time, as statistical methods for prediction commonly produce probabilities for the occurrence of a specific event. As is shown in the diagram below, a decision is made at a specific point in time to protect assets, or not to protect assets. The forecast event then either occurs or does not occur. Costs are simply the value in the currency chosen. These costs are usually given in dollar values, but can be man hours, days of delays, or some other currency. Losses are usually expressed in dollars, and represent the replacement cost of the item.

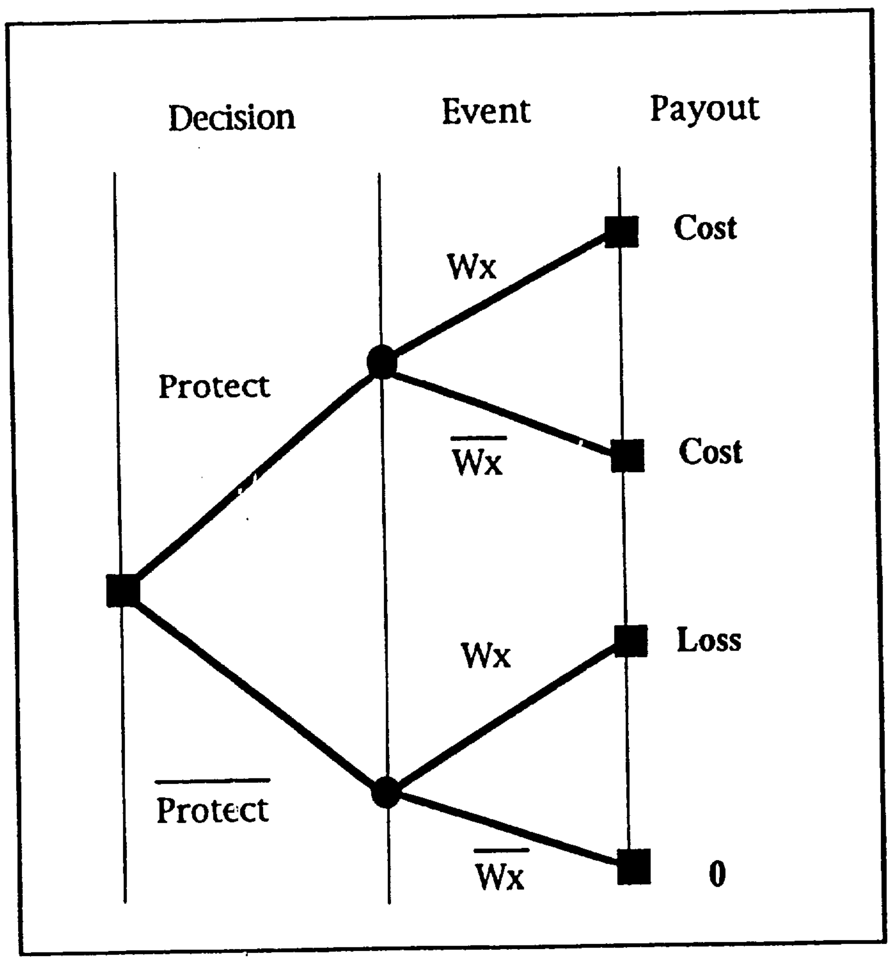


Diagram 1. Simplified Cost versus Loss decision making tree

The decision is made as follows:

$$If\ p < \frac{C}{L}\ then\ do\ not\ protect$$

where p is the forecast probability of the meteorological event, C is the cost of protection, and L represents the loss value. Should the p be greater than the cost/loss ratio, then one would protect.

In the case of protecting the shuttle, assume that C/L is equal to 1/500, a somewhat arbitrary figure[2]. Studies[3] have shown that hurricanes have passed within 75 nautical miles on 26 occasions since 1886. If one considers hurricane season to last from 1 June through November (183 days) , and hurricanes to affect the area for only one day each, the probability of being hit on any one day in hurricane season is found to be 26 in 19398. In a probability to cost/loss comparison:

$$0.0013403 < 0.002$$

Thus, one would not protect on a day to day basis since the probability of the event is less the cost/loss ratio[4]. Had the probability been greater that the C/L, it would be unreasonable to expect that rollback should occur on each day of the season, there would be no launches at all!

2.2 SEQUENTIAL DECISIONS

The record of tropical storm activity from 1886 through 1991 shows that of 970 storms, 614 reached hurricane strength. This averages out to about 9 storms per year, with nearly 6 reaching hurricane strength. Rarely are hurricanes dependent upon one another, and their existence can be thought of as independent sequential events. The decision making tree for more than one event becomes complicated after very

---

2 Although not unreasonable this number is optimistic at best. In the currency of days: one day of lost processing time to 500 days to replace a shuttle.

3 NOAA TM NWS NHC 38, reprinted Aug 91.

4 In reality, it is unreasonable to expect that a hurricane with winds in excess of 74.5 knots would spontaneously appear close enough to do damage over the course of 24 hours, so this daily probability is really only useful in determining climatological probabilities, and works out to about a one in four chance of having a hurricane in any given year.

few iterations. Thus, the example of such a tree is shown below with associated payouts for only two events.

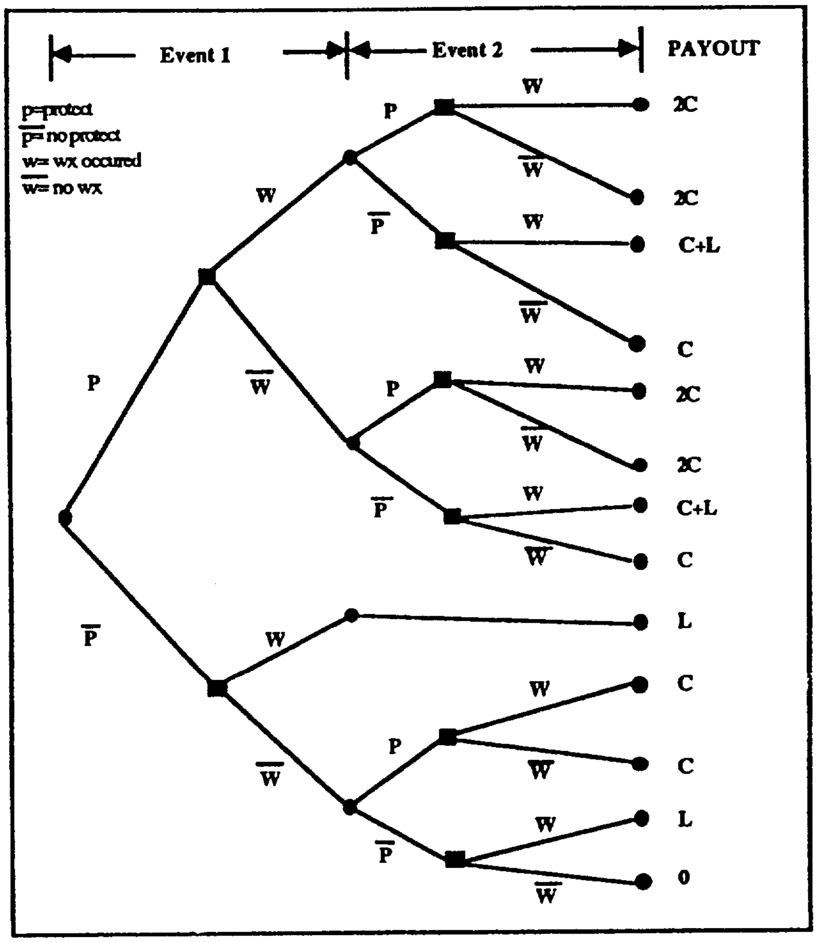


Diagram 2. Decision tree for 2 events

For each additional event, another four pronged branch is added to the tree at all locations where loss does not occur. For n events, the number of final nodes would be:

$$\textit{number of nodes} = 3^n + \sum_{x=1}^{n} x^2$$

where the summation gives the number of those nodes which end in losses. Only one node yields no payouts, occurs when you never protect and never have the occurrence.

This decision tree is based upon having but one orbiter on a launch pad at a time, and assumes that after a loss shuttle operations would cease. In reality this is probably the case, but it could be argued that since the loss of the orbiter is know to be due to a management decision as opposed to an engineering flaw, operations might continue shortly thereafter. Were this scenario to take place, the number of nodes would increase to:

$$number\ of\ nodes = 4^n$$

We shall use the first example to generate the decision probabilities. This is done by using 'backward induction' from the endpoint to the initial condition. In our example, consider the case where the p>(C/L). On day 2, the decision maker should protect. Thus, if the decision maker decides to protect on day 1, the cost for the two days is 2C. If the decision maker does not protect on day 1, he incurs a risk of pL of the loss and saves -(1-p)C by not protecting. Thus, his expected payoff (or savings) will be:

$$-pL-(1-p)C$$

Remember, L and C are payouts, and are negative numbers which makes the overall number positive[5]. Thus, the decision maker should protect on day 1 if:

$$2C > -pL-(1-p)C$$

Which reduces to:

$$p > C/(L-C)$$

The manager should protect on the first day based on is new relationship. In tabular form the threshold values for each of the possible situations are as follows.

Case	Day 1	Day 2	Payoff	Optimal when
1	p	p	-2C	$C/(L-C)<p\leq 1$
2	p	not p	-C-pL	never[6]
3	not p	p	-pL-(1-p)C	$C/L<p<C/(L-C)$
4	not p	not p	$-[1-(1-p)^2]L$	$p\leq C/L$

5 Unless the probability of the event is .5 in which case one should always protect.

6 Since the payoff for case 3 is less than or equal the payoff of case 2 for all cases, case 2 is never optimal.

If we assume values bases for cost and loss based upon dollar values of \$3 billion for loss and as mentioned above and \$2 million for cost of rollback per day with five lost days,[7] for a cost of \$10 million. If the probability of strike is .01 the table looks like:

Case	Day 1	Day 2	Payoff	Optimal when
1	.01	.01	20	$0.0033444<p\leq1$
2	.01	.99	40	never[8]
3	.99	.01	39.9	$0.003333<p<0.0033444$
4	.99	.99	59.7	$p\leq0.0033333$

It becomes obvious that because of the large disparity between the cost of protection versus loss, protection should be implemented whenever forecasts indicate strike probability is greater than some number less than one. As the cost of protection increases, however, threshold forecast probabilities also increase, albeit slowly. However, using the above tables in a spreadsheet form, one could easily tailor the optimization given the revised cost and loss values based upon scheduled launches and payloads, and even the two shuttle situation.

7 Two days for rollback, one day during storm, and two days to get back to initial conditions. Best case scenario.

8 Since the payoff for case 3 is less than or equal the payoff of case 2 for all cases, case 2 is never optimal.

SECTION III COMPUTER TRACK ANALYSIS

Another study was initiated using historical data of hurricane and tropical storm tracks. These data were acquired last summer, and comprise track data for storms from 1886 through 1991. Storm data is in ASCII text format, and contains six hourly information containing position, wind, and sea level pressure (Appendix 1).

One goal set last year was to create the ability to examine these data in a graphical form. It was felt that by selecting specific storms similar in nature to the current storm, a probability study on forward motion could be undertaken which could then be used along with the NHC warning[9] to afford local forecasters another tool for determination of landfall. In addition, the computer access to track data would allow for numerous additional courses of study. The source code for these programs are contained in the appendices and program documentation is integral to the source code.

3.1 PROGRAM DEVELOPMENT AND REQUIREMENTS

Programming was written in Borland's Turbo Pascal on an MS-DOS laptop computer. The goal was to create executable programs that would be easily transportable to that type of machine, and to the extent of the programs written, this goal has been successfully met. Machine requirements are minimal, an MS-DOS machine with one megabyte RAM, and VGA display. As Borland also has Pascal available for the Apple Macintosh machine, the source code can be transferred to a Mac, then compiled and run. Since data files are also transferable (being data files) no significant revision of source code is anticipated.

One note about the programming environment. It was chosen because it satisfied several criterion. First, it was easy to edit code in the Borland environment. Conversations with computer programmers at the center pointed out that fact. Secondly, compilation linking and running are accomplished very easily with syntax errors found and easily corrected, as the environment places the cursor at the error in the edit mode for immediate correction. Finally, the graphical capabilities were greatly enhanced by the extensive library of commands and features available

---

9 NHC has recently incorporated a method whereby historical forecasts of similar storms are used in obtaining forecast guidance, and help in assessing the confidence level in the current forecast.

in the environment. I found it extremely easy to write code to accomplish my task in reasonable time. Having no Pascal experience (but speaking several other languages) the rather steep learning curve was possible only due to the user friendliness of the language.

3.2 DATA AND SUBROUTINES

Each program uses one or both of two data files. One, distributed through shareware, is a database which originally was produced by the Central Intelligence Agency. The documentation accompanying that program, PLOTMWDB, contains an interesting history of the data. It also provides a scheme for unpacking and reading data. Generally, data points are described by type (coastline, river, island, etc.) and coordinates given in minute form. Conversion of these data to pixel address points for screen display was a particularly satisfying success. The procedure named 'MAKEMAP' accomplishes this.

The storm track file, as mentioned, is a text file. Its conversion to integer and real number data proved to be another interesting challenge which was overcome primarily with the use of three procedures which convert two, three and four sequential text characters into integer format for use in the storm plotting routines. That conversion to pixel points makes use of several procedures, one which converts from decimal degrees to minutes, determines if the point is visible in the defined visible screen window. If so, it is plotted, if not, it is discarded. Each visible storm fix is plotted along with adjoining line segments with different colored circles at the fix points representing storm strengths, namely, depression, storm and hurricane. Minor code modifications would permit additional differentiation, for example, based upon category level. The storm data acquisition and plotting is accomplished with a procedure named 'GETSTORM', and appears in several programs as does the map making program.

Another interesting problem was in determining the best way to erase a storm track in the storm selection program. Using the 'exclusive or' feature in the graphing package, writing the screen with the track twice effectively returns it to the original color, in effect erasing it. This required reading a storm data file, and saving it to a temporary location so that it could be reread and plotted. Then, calling a write procedure to plot the storm track, and then calling the write procedure again to rewrite it which in effect erased it. This method was required by the sequential read nature of the original storm track data file and the inability to back step easily.

The following is a brief summary of the two programs which evolved over the course of the summer. The first selects data, the second plots selected years.

3.3 PROGRAM PICKTRAK.PAS

As the name implies, this program was created to allow the user to select the storms of his choice. In it's present form, it requests a storm data file name (source file) and then asks for the output file name. After generating a map of the area it sequentially plots the storm track, gives general information concerning that storm, and asks if the user wants to include it in the output file. Any keystroke but 'Y' (not case sensitive) bypasses the storm. This program has the potential of being modified to accomplish many different selection schemes. Yearly data, storms of specified strength, storms passing within x miles of a location, probability distribution functions based upon historical data may all be fairly easily woven into the code.

3.4 PROGRAM PLOTYR.PAS

Designed to allow the user to examine storm data for a given year. On execution, the program requests a file name which contains the storm data which the forecaster is interested in displaying. It then asks for a forward storm speed. This velocity is used to generate event circles which are centered about KSC, and represent decision points for (from center outward) 8,12,24,36,48 and 72 hours. These correspond to the rollback timeline event critical points (chart 3). The program then shifts to graphics mode, generates the map of the region, and plots the storm tracks for the selected data file. This program has the potential to allow selection of specific storm criterion with minor revisions in the source code.

IV. CONCLUDING REMARKS

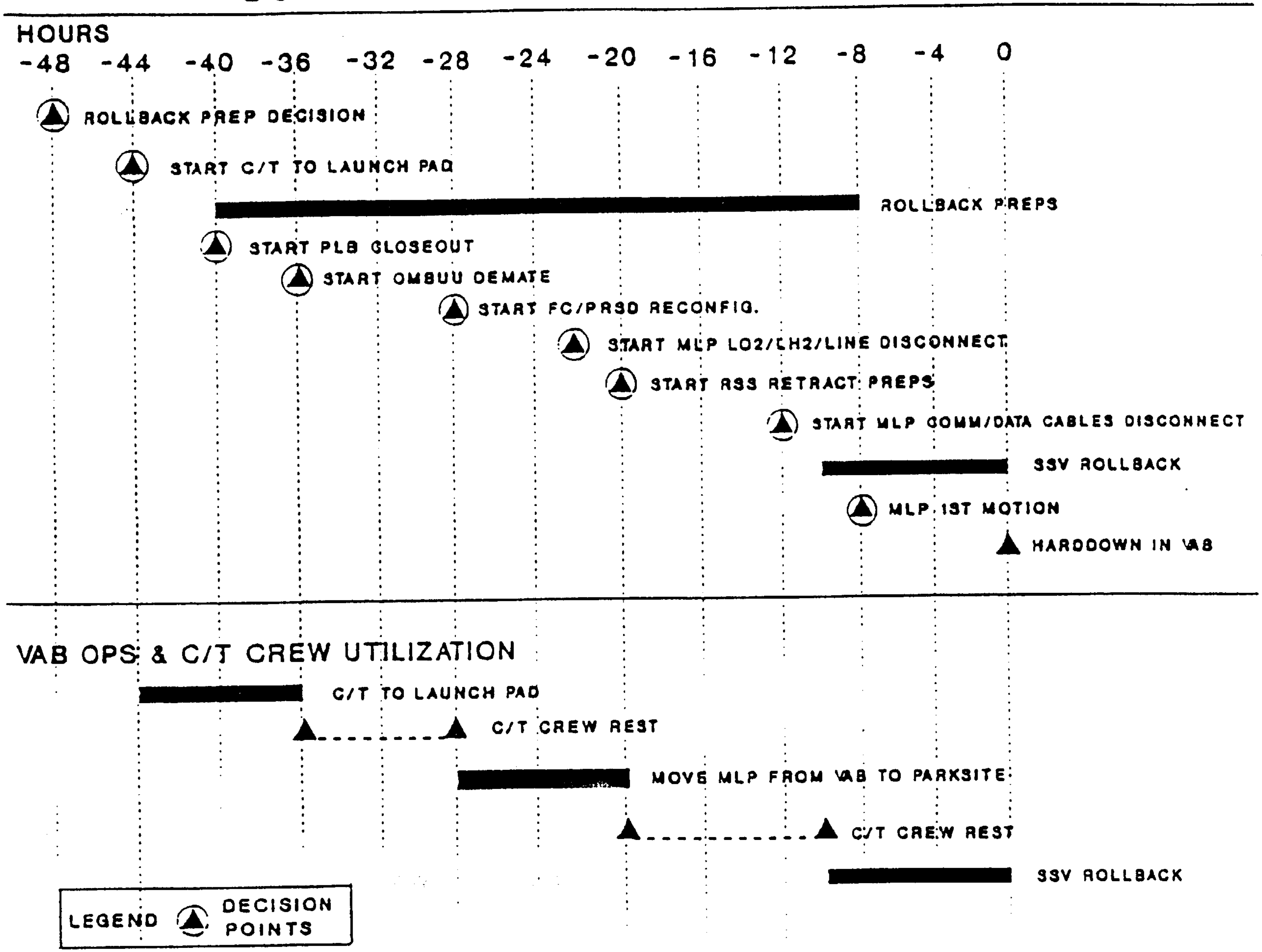
Determination of a fixed probability value for making a rollback decision varies with the number of storms that are forecast to occur at the center, and is very dependent upon the initial values one places on the cost of operations and the loss of the vehicle. Cost is difficult at best to quantify, because of the variability of situations that might exist; one shuttle vs. two, recently moved shuttle, processing accomplished, and even availability of space in the VAB add varying degrees of complexity to the problem. Loss, typically thought of as strictly a dollar value, is again fogged by the value of the space shuttle in the eyes of the nation, and its value in terms of a national resource. Replacement cost might be prohibitive in this period of economic belt tightening, and the adverse effect of such a loss on NASA's reputation and effectiveness would be disastrous. For this reason, the author feels that rollback decisions should be made at a low probability of strike, on the order of 1 or 2 percent.

The time to make those decisions can be fairly easily determined from the NHC warnings. Forward speed times 48 gives the decision circle to use.

The computer packages developed through the course of the study can be used by forecasters to examine more closely the historical data. They also provide a strong departure point for additional studies that might improve hurricane forecasting. Some of those would be the development of a statistical analysis of selected historical storm tracks. For example, one of the tasks not accomplished this summer is a 'forward scatter' capability based on selected storms. It was felt that if a significant number of storms from the historical record were analyzed, probabilities based upon behavior might become evident. For example, if a storm 300 Nautical miles due east were to begin tracking from a heading of 270^o to 290^o, what percent of similar storms then turned back westward? In other words, what are the chances that the current storm will 'get us'? This methodology could be verified by using real storms which could be shifted so that their track passed within a specified number of miles of the center. The value of the success of this method would allow managers to live with a higher probability of strike.

APPENDICES

SSV ROLLBACK MILESTONE SCHEDULE



```
62410 05/17/1970 M=11  1 SNBR= 688 ALMA        XING=0
62420 05/17*  0   0   0     0*  0   0   0    0*  0   0   0     0*115 790  25     0*
62430 05/18*117 792  25     0*121 797  25    0*123 801  25 1007*125 805  25     0*
62440 05/19*130 810  25     0*135 815  25    0*140 820  25 1005*145 825  25     0*
62450 05/20*155 825  35     0*162 823  35    0*168 819  65    0*172 816  70   993*
62460 05/21*180 810  55     0*182 803  50    0*185 802  45    0*190 800  40   998*
62470 05/22*196 801  35 1004*193 806  30    0*191 811  30    0*190 818  30  1007*
62480 05/23*192 827  30     0*198 835  25    0*210 839  25    0*219 840  25     0*
62490 05/24*230 840  25     0*240 840  25    0*252 840  25 1008*268 839  25     0*
62500 05/25*283 836  25     0*295 829  25    0*306 822  25    0*318 816  25     0*
62510 05/26*326 812  25 1005*333 807  25    0*348 799  25    0*357 787  25     0*
62520 05/27E365 770  25 1003E370 755  25    0*  0   0   0     0*  0   0   0     0*
62530 HR
```

```
PROGRAM PICKTRAK (INPUT,OUTPUT);

USES Graph,DOS,CRT;

{*******************INITIALIZATIONS**************************}

TYPE
  POINT = RECORD                  {Micro World Data Base II input record}
            CODE  : INTEGER;
            LAT   : INTEGER;
            LON   : INTEGER
          END;


VAR

    NEWPOINT          : POINT;
    INFILEPT          : FILE OF POINT;
    INFILEPT_NAME     : STRING[14];
    INFILE            : FILE OF CHAR;
    IN_FILE           : STRING[14];
    OUTFILE           : FILE OF CHAR;
    OUT_FILE          : STRING[14];


   HDRFILE            : TEXT;
   DATA               : ARRAY[1..30,1..81] OF CHAR;
   HEADLINE           : STRING[81];
   DUMMY              : STRING[80];    {dummy variable for keyboard input}
   TEMP               : STRING[53];
   C,D                : CHAR;
   VEL                : REAL;
   MaxX               : INTEGER;       {max x screen size for graphics}
   MaxY               : INTEGER;       {max y screen size for graphics}
   GDVR               : INTEGER;
   GRMODE             : INTEGER;
   PC                 : INTEGER;
   i,j,k,m,n,r        : INTEGER;
   T,U,V,X,JJ         : INTEGER;
   minlon             : INTEGER;
   OLDLAT,OLDLON      : INTEGER;
   WINLAT,WINLON      : INTEGER;
   HERELAT,HERELON    : INTEGER;

   MaxColor           : WORD;          {max color value for graphics}

   MAP_FILE           : FILE OF POINT; {symbolic name for input map data file}
   FILE_NAME          : STRING[16];    {stores map file name input by user}

{   MAX_LAT    : REAL;         {stores north, south,}
{   MIN_LAT    : REAL;           {east, and west limits of area}
{   MAX_LON    : REAL;             {to be displayed - }
{   MIN_LON    : REAL;               {input by user}

{   WINXMAX,WINXMIN: REAL;     {max and min latitude and longitude}
{   WINYMAX,WINYMIN: REAL;       {values used in window generating routine}

    MO,DY,YR            : INTEGER;
    NMBR,LIFE           : INTEGER;
    NAME                : STRING[10];
    LT                  : ARRAY[1..100] OF INTEGER;
    LN                  : ARRAY[1..100] OF INTEGER;
    WN                  : ARRAY[1..100] OF INTEGER;
    SP                  : ARRAY[1..100] OF INTEGER;


   VISIBLE,OLDVIS :BOOLEAN;
{****************************  INITIALIZE   ************************}

PROCEDURE Initialize;
begin

{initialize the graphics}
     DetectGraph(GDVR,GRMODE);
     InitGraph(GDVR,GRMODE,' ');

{gets max x and y values for screen}

          MaxX        := GetMaxX;
          MaxY        := GetMaxY;
          MaxColor    := GetMaxColor;
          SetViewPort(0,0,MaxX,MaxY,ClipOn);
          SetLineStyle(Solidln,0,NormWidth);
          SetColor(MAXCOLOR);
end;
{************************  ERASEBLOCK   ******************************}

PROCEDURE ERASEBLOCK(LEFT,TOP,RIGHT,BOTTOM:INTEGER);

VAR
   X,Y,Z  : INTEGER;
   BEGIN
        Z:=GETBKCOLOR;
        FOR X:= LEFT TO RIGHT DO
            FOR Y:=TOP TO BOTTOM DO
                PUTPIXEL(X,Y,Z);
   END;
{*************************  BORDER   ****************************}

PROCEDURE border;              {border of screen output}


     begin
          SetColor(15);
          SetLineStyle(Solidln,0,ThickWidth);
          Rectangle(0,0,maxx,maxy-120);          {draw thick border in whit.
          Setcolor(RED);
          SetLineStyle(Solidln,0,Normwidth);     {around thin border in grey}
```

```
        Rectangle(3,3,maxx-3,maxy-123);
        SetColor(white)
    end;
{****************** LATLONCON *******************************}

Procedure latloncon (y,x:longint);
        {This function converts from map minute coordinates to
              window coordinates for plotting lat/lon on the screen
                    and determines if it is in the visible window}
        begin

            IF Y<300 THEN VISIBLE:= FALSE      {minlat = equator}
            ELSE IF Y>2400 THEN VISIBLE:= FALSE {maxlat = 40N}
            ELSE IF X<-6600 THEN VISIBLE:= FALSE {maxlon = 110W}
            ELSE IF X>-1199 THEN VISIBLE:= FALSE {minlon = 20W}
            ELSE VISIBLE:=TRUE;

            winlon:= (x+5500) div 5;
            winlat:= ((y-2100) div -5);
        end;
{************************** GRIDLINES ************************}

Procedure gridlines;
    BEGIN
    SetColor(black);                {linecolor is dark gray}
    SetLineStyle (1,0,1);          {line style is dotted}
        m := -5000;                 {dummy lon for latlonconv routine}
        for i := 1 to 8 do
            begin
                n:=i*300;                   {figures winlat values from }
                latloncon(n,m);              {5N to 40N}
                if visible then
                begin
                    MoveTo(0,Winlat);
                    LineTo(639,Winlat);        {draws the line}
                end
            end;
        n:=650;                      {dummy lat for latlonconv routine}
        for i := 1 to 20 do
            begin
                m:= -900+(-i*300);
                latloncon(n,m);              {figures winlon values from}
                MoveTo(Winlon,0);            {20W to 115W}
                Lineto(winlon,479);          {draws line}
            end;
        end;
{**************************** ENDOFLINE ****************************}

PROCEDURE ENDOFLINE;
BEGIN
        READ(INFILE,C);
        IF C<>CHR(10) THEN
            BEGIN
                REPEAT
                  READ(INFILE,C);
                UNTIL C=CHR(10);
            END;
END;
{**************************** MAKETWO ********************************}

PROCEDURE MAKETWO(T,U:INTEGER);

VAR
    P,Q  : INTEGER;

BEGIN

        IF HEADLINE[T]=' ' THEN P:=0
           ELSE P:=(INTEGER(HEADLINE[T])-48);
        IF HEADLINE[U]=' ' THEN Q:=0
           ELSE Q:=(INTEGER(HEADLINE[U])-48);
        X:=P*10+Q;;
END;
{**************************** MAKETHREE ********************************}

PROCEDURE MAKETHREE(T,U,V:INTEGER);

VAR
    P,Q,R      : INTEGER;

BEGIN
        IF HEADLINE[T]=' ' THEN P:=0
           ELSE P:=(INTEGER(HEADLINE[T])-48);
        IF HEADLINE[U]=' ' THEN Q:=0
           ELSE Q:=(INTEGER(HEADLINE[U])-48);
        IF HEADLINE[V]=' ' THEN R:=0
           ELSE R:=(INTEGER(HEADLINE[V])-48);
        X:=P*100+Q*10+R;
END;
{**************************** MAKEFOUR ********************************}

PROCEDURE MAKEFOUR(T,U,V,W:INTEGER);

VAR
    P,Q,R,S     : INTEGER;

BEGIN
        IF HEADLINE[T]=' ' THEN P:=0
           ELSE P:=(INTEGER(HEADLINE[T])-48);
        IF HEADLINE[U]=' ' THEN Q:=0
           ELSE Q:=(INTEGER(HEADLINE[U])-48);
        IF HEADLINE[V]=' ' THEN R:=0
           ELSE R:=(INTEGER(HEADLINE[V])-48);
        IF HEADLINE[W]=' ' THEN S:=0
           ELSE S:=(INTEGER(HEADLINE[W])-48);
        X:=P*1000+Q*100+R*10+S;
END;
```

```
{*************************** WRITEHEADER ****************************}

PROCEDURE WRITEHEADER;

BEGIN
        FOR I:=1 TO 52 DO
        WRITE(HDRFILE,TEMP[I]);
        WRITELN(HDRFILE,TEMP[53]);
END;
{*************************** WRITEDATA ******************************}

PROCEDURE WRITEDATA(X:INTEGER);

VAR
   I,J     : INTEGER;

BEGIN
     FOR I:=1 TO X DO
     BEGIN
          FOR J:=1 TO 79 DO WRITE(HDRFILE,DATA[I,J]);
          WRITELN(HDRFILE,DATA[I,80]);
     END;
     WRITELN(HDRFILE,'**********');
END;

{************************* PLOTPOINT ***************************}

PROCEDURE PLOTPOINT(X,Y:INTEGER);

BEGIN
     SetLineStyle(Solidln,0,ThickWidth);
     LATLONCON(X,Y);
     IF VISIBLE THEN
        BEGIN
             IF OLDVIS THEN
                BEGIN
                     MOVETO(OLDLON,OLDLAT);
                     LINETO(WINLON,WINLAT);
                END;
             OLDLAT:=WINLAT;
             OLDLON:=WINLON;
             OLDVIS:=TRUE;
        END
        ELSE
        BEGIN
             OLDVIS:=FALSE;
        END;
END;

{************************* PLOTKSC ***************************}

PROCEDURE PLOTKSC;

VAR
     NM      : INTEGER;

BEGIN
     SETFILLSTYLE(SOLIDFILL,RED);
     HERELAT:=ROUND(28.3833*60);
     HERELON:=ROUND(80.5833*-60);
     LATLONCON(HERELAT,HERELON);
     FILLELLIPSE(WINLON,WINLAT,2,2);  {PLOTS KSC AS A RED DOT}
     HERELON:=WINLON;
     HERELAT:=WINLAT;

     NM:=12;                   {NM=1 DEG = 60NM}
     SETCOLOR(WHITE);
     CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*9/50));   {PLOTS EVENT CIRCLES}
     CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*12/60));  {DISTANCES: 100,200,AND 300.
     CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*24/60));
     CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*36/60));
     CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*48/60));
     CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*72/60));
END;
{**************************** PLOTTRACK ******************************}

PROCEDURE PLOTTRACK(X:INTEGER);

BEGIN
     VISIBLE:=FALSE;
     OLDVIS:=FALSE;
     X:=X*4;
     FOR I:=1 TO X DO
         BEGIN
              PLOTPOINT(LT[I]*6,LN[I]*-6);
         END;
END;
{****************************** CLRTRK *******************************}

PROCEDURE CLRTRK;

VAR
   I : INTEGER;

   BEGIN
        FOR I:=1 TO 100 DO
        BEGIN
             LT[I]:=0;
             LN[I]:=0;
        END;
   END;
{****************************** GETSTORM *****************************}

PROCEDURE GETSTORM;

 BEGIN
```

```
            J:=(I-1)*17;
            MAKETHREE(13+J,14+J,15+J);
              LT[I+((JJ-1)*4)]:=X;
            MAKETHREE(17+J,18+J,19+J);
              LN[I+((JJ-1)*4)]:=X;
            MAKETHREE(21+J,22+J,23+J);
              WN[I+((JJ-1)*4)]:=X;
              IF WN[I]<35 THEN PC:=3
              ELSE IF WN[I]<65 THEN PC:=14
              ELSE PC:=60;
            MAKEFOUR(25+J,26+J,27+J,28+J);
              SP[I+(JJ-1*4)]:=X;
        END;
        FOR I:=1 TO 10 DO
            DATA[JJ+1,I]:='*';
      END;
      ENDOFLINE;
      PLOTTRACK(K);
      OUTTEXTXY(0,410,'DO YOU WANT THIS TRACK? (Y or N) ');
      D:=READKEY;
      OUTTEXTXY(264,410,D);
      IF (D=CHR(89)) OR (D=CHR(121)) THEN
         BEGIN
              WRITEHEADER;
              WRITEDATA(K);
         END;
      PLOTTRACK(K);
      CLRTRK;
      ERASEBLOCK(0,380,300,450);
   END;
CLOSE(INFILE);
CLOSE(HDRFILE);
END;

{**************************** MAKEMAP ****************************}

PROCEDURE MAKEMAP;


BEGIN
     ASSIGN(INFILEPT,'NEWPOINT.DAT');
     RESET(INFILEPT);
   WHILE NOT EOF(INFILEPT) DO
    BEGIN
      READ(INFILEPT, NEWPOINT);
      WITH NEWPOINT DO
             IF CODE > 5 THEN
           BEGIN
             LATLONCON(LAT,LON);
             IF VISIBLE THEN
                BEGIN
                  MOVETO(winlon,winlat);
                  oldlat:=winlat;
                  oldlon:=winlon;
                  oldvis:=visible;
                END
```



```
ASSIGN(INFILE,IN_FILE);
ASSIGN(HDRFILE,OUT_FILE);
RESET(INFILE);
REWRITE(HDRFILE);

WHILE NOT EOF(INFILE) DO
      BEGIN
      OLDVIS:=FALSE;
      VISIBLE:=FALSE;
           FOR I:=1 TO 52 DO
                 BEGIN
                      READ(INFILE,HEADLINE[I]);
                      TEMP[I]:=HEADLINE[I];
                 END;
           READ(INFILE,C);
           HEADLINE[53]:=C;
           TEMP[53]:=C;
           ENDOFLINE;
             MAKETWO(20,21);
               K:=X;
             MAKEFOUR(31,32,33,34);
               NMBR:=X;
             MAKETWO(7,8);
               MO:=X;
             MAKETWO(10,11);
               DY:=X;
             MAKEFOUR(13,14,15,16);
               YR:=X;
               OUTTEXTXY(0,380,'STORM START DATE: ');
               FOR I:=7 TO 16 DO
               OUTTEXTXY(90+8*I,380,HEADLINE[I]);
               OUTTEXTXY(0,390,'STORM NUMBER: ');
               FOR I:=1 TO 4 DO
               OUTTEXTXY(90+8*I,390,HEADLINE[I+30]);
               OUTTEXTXY(0,400,'THIS STORM IS');

             FOR I:=1 TO 10 DO
                BEGIN
                NAME[I]:=HEADLINE[35+I];
                OUTTEXTXY(110+8*I,400,NAME[I]);
                END;
      FOR JJ:=1 TO K DO
             BEGIN
               FOR I:=1 TO 81 DO
               BEGIN
                    READ(INFILE,C);
                    HEADLINE[I]:=C;
                    DATA[JJ,I]:=C;
               END;
                    READ(INFILE,C);
               MAKETWO(7,8);
                 MO:=X;
               MAKETWO(10,11);
                 DY:=X;
               FOR I:=1 TO 4 DO
               BEGIN
```

```
          ELSE CLDVIS:=FALSE

        END
      ELSE
        BEGIN
          LATLONCON(LAT,LON);
          IF NOT VISIBLE THEN
             OLDVIS :=FALSE
          ELSE IF NOT OLDVIS THEN
               BEGIN
                    MOVETO(WINLON,WINLAT);
                    OLDLAT:=WINLAT;
                    OLDLON:=WINLON;
                    OLDVIS:=TRUE;
               END
          ELSE
               BEGIN
                    LINETO(WINLon,WINLat);
                    oldlat:=winlat;
                    oldlon:=winlon;
                    oldvis:=TRUE;
               END
            END
      END;
      OLDVIS:=FALSE;
   SETFILLSTYLE(SOLIDFILL,CYAN);
   FLOODFILL(600,100,WHITE);
   FLOODFILL(10,300,WHITE);
   CLOSE(INFILEPT);
   LATLONCON(610,-4300);
   FLOODFILL(WINLON,WINLAT,WHITE);
   SETFILLSTYLE(SOLIDFILL,CYAN);
   LATLONCON(1620,-4850);
   FLOODFILL(WINLON,WINLAT,WHITE);
   LATLONCON(700,-5130);
   FLOODFILL(WINLON,WINLAT,WHITE);

END;
{************************* MAIN PROGRAM *************************}

BEGIN

    CLRSCR;
    WRITELN('WHAT IS THE NAME OF YOUR STORM DATA FILE?');
    READLN(IN_FILE);
    WRITELN('WHERE WOULD YOU LIKE TO SAVE YOUR SELECTED STORMS?');
    READLN(OUT_FILE);
    {WRITELN('HOW FAST IS THE CURRENT STORM MOVING?');
    READLN(VEL);
    IN_FILE:='TEST.DAT';}
    INITIALIZE;
    BORDER;
    SETFILLSTYLE(SOLIDFILL,GREEN);          {fills screen with GREEN }
    FLOODFILL(100,100,WHITE);
    oldlat:=0;
    oldlon:=0;
    oldvis:=false;

        MAKEMAP;
        GRIDLINES;
        BORDER;
        PLOTKSC;
        SETWRITEMODE(XORPUT);
        GETSTORM;
        OUTTEXTXY(1,400,'THAT COMPLETES THE DATA. PRESS RETURN TO EXIT');
        READLN(DUMMY);
        CloseGraph;
END.    {MAIN PROGRAM}
```

```
PROGRAM PLOTYR (INPUT,OUTPUT);

{This program will plot the storm tracks from data stored in a file
which the user may select. The program will prompt you for a year}

USES Graph,DOS,CRT,PRINTER;

{*****************INITIALIZATIONS*************************}

CONST
   ESC = #$1B;

TYPE
   POINT = RECORD                  {Micro World Data Base II input record}
             CODE  : INTEGER;
             LAT   : INTEGER;
             LON   : INTEGER
          END;

CONST
    OUT=10;

VAR

     NEWPOINT      : POINT;
     INFILEPT      : FILE OF POINT;
     INFILEPT_NAME : STRING[14];
     INFILE        : FILE OF CHAR;
     IN_FILE       : STRING[14];

   DUMMY       : STRING[80];   {dummy variable for keyboard input purposes}

   C           : CHAR;         {used for reading in data from storm file}
   VEL         : REAL;         {velocity}
   MaxX        : INTEGER;      {max x screen size for graphics}
   MaxY        : INTEGER;      {max y screen size for graphics}
   GDVR        : INTEGER;      {graphics driver}
   GRMODE      : INTEGER;      {graphics mode}
   PC          : INTEGER;      {for coloring position dots}
   i,j,k,a,n,r: INTEGER;
   T,U,V,X,JJ : INTEGER;
   OLDLAT,OLDLON   : INTEGER;  {used in the position routine}
   WINLAT,WINLON   : INTEGER;  {lat and lon for position}
   HERELAT,HERELON: INTEGER;   {lat/lon for KSC}

   MaxColor    : WORD;         {max color value for graphics}

   MAP_FILE    : FILE OF POINT; {symbolic name for input map data file}
   FILE_NAME   : STRING[16];    {stores map file name input by user}
   WINXMAX,WINXMIN: REAL;       {max and min latitude and longitude}
   WINYMAX,WINYMIN: REAL;       {values used in window generating routine}
                                                {variables for storm}
    MO,DY,YR              : INTEGER;             {data, selfevident}
    NMBR,LIFE             : INTEGER;
    NAME                  : STRING[10];

    LT                     : ARRAY[1..100] OF INTEGER;
    LN                     : ARRAY[1..100] OF INTEGER;
    WN                     : ARRAY[1..100] OF INTEGER;
    SP                     : ARRAY[1..100] OF INTEGER;

    HEADLINE               : STRING[80];      {used in data read routine}
    NEWLINE                : STRING[80];

    VISIBLE,OLDVIS         :BOOLEAN;          {flags to point points}

{****************************  INITIALIZE  ************************}

PROCEDURE Initialize;
begin

{initialize the graphics}
     DetectGraph(GDVR,GRMODE);
     InitGraph(GDVR,GRMODE,' ');

{gets max x and y values for screen}

          MaxX         := GetMaxX;
          MaxY         := GetMaxY;
          MaxColor     := GetMaxColor;
          SetViewPort(0,0,MaxX,MaxY,ClipOn);
          SetLineStyle(Solidln,0,NormWidth);
          SetColor(MAXCOLOR);
end;
{*********************** BORDER  ******************************}

PROCEDURE border;                 {border of screen output}


     begin
          SetColor(15);
          SetLineStyle(Solidln,0,ThickWidth);
          Rectangle(0,0,maxx,maxy-120);          {draw thick border in white}
          Setcolor(RED);
          SetLineStyle(Solidln,0,Normwidth);     {around thin border in red}
          Rectangle(3,3,maxx-3,maxy-123);
          SetColor(white)
     end;
 {****************** LATLONCON  ******************************}

 Procedure latloncon (y,x:longint);
         {This function converts from map minute coordinates to
               window coordinates for plotting lat/lon on the screen
                      and determines if it is in the visible window}
         begin

              IF Y<300 THEN VISIBLE:= FALSE      {minlat = equator}
              ELSE IF Y>2400 THEN VISIBLE:= FALSE {maxlat = 40N}
              ELSE IF X<-6600 THEN VISIBLE:= FALSE {maxlon = 110W}
```

```
            ELSE IF X>-1199 THEN VISIBLE:= FALSE {minlon = 20W}
            ELSE VISIBLE:=TRUE;

            winlon:= (x+6000) div 5;
            winlat:= ((y-2100) div -5);
       end;
{************************** GRIDLINES *************************}

Procedure gridlines;
   BEGIN
   SetColor(black);                {linecolor is dark gray}
   SetLineStyle (1,0,1);           {line style is dotted}
      m := -5000;                  {dummy lon for latlonconv routine}
      for i := 1 to 3 do
          begin
              n:=i*300;                    {figures winlat values from }
              latloncon(n,m);              (5N to 40N)
              if visible then
              begin
                  MoveTo(0,Winlat);
                  LineTo(639,Winlat);      {draws the line}
              end
          end;
      n:=650;                      {dummy lat for latlonconv routine}
      for i := 1 to 20 do
          begin
              m:= -900+(-i*300);
              latloncon(n,m);              {figures winlon values from}
              MoveTo(Winlon,0);             (20W to 115W)
              Lineto(winlon,479);           {draws line}
          end;
    end;
{**************************** ENDOFLINE ******************************}

PROCEDURE ENDOFLINE;                {used in data read to scan thru extraneous}
BEGIN                               {trash at end of data lines}
        READ(INFILE,C);
        IF C<>CHR(10) THEN
           BEGIN
                REPEAT
                  READ(INFILE,C);
                UNTIL C=CHR(10);
           END;
END;
{**************************** MAKETWO ********************************}

PROCEDURE MAKETWO(T,U:INTEGER);     {makes a two digit integer from two}
                                    {sequential text characters}
VAR
   P,Q  : INTEGER;

BEGIN

        IF HEADLINE[T]=' ' THEN P:=0
           ELSE P:=(INTEGER(HEADLINE[T])-48);
        IF HEADLINE[U]=' ' THEN Q:=0
           ELSE Q:=(INTEGER(HEADLINE[U])-48);
        X:=P*10+Q;;
END;
{**************************** MAKETHREE ********************************}

PROCEDURE MAKETHREE(T,U,V:INTEGER);
                                          {makes three digit integer}
VAR
   P,Q,R      : INTEGER;

BEGIN
        IF HEADLINE[T]=' ' THEN P:=0
           ELSE P:=(INTEGER(HEADLINE[T])-48);
        IF HEADLINE[U]=' ' THEN Q:=0
           ELSE Q:=(INTEGER(HEADLINE[U])-48);
        IF HEADLINE[V]=' ' THEN R:=0
           ELSE R:=(INTEGER(HEADLINE[V])-48);
        X:=P*100+Q*10+R;
END;
{**************************** MAKEFOUR **********************************}

PROCEDURE MAKEFOUR(T,U,V,W:INTEGER);
                                          {makes four digit integer}
VAR
   P,Q,R,S     : INTEGER;

BEGIN
        IF HEADLINE[T]=' ' THEN P:=0
           ELSE P:=(INTEGER(HEADLINE[T])-48);
        IF HEADLINE[U]=' ' THEN Q:=0
           ELSE Q:=(INTEGER(HEADLINE[U])-48);
        IF HEADLINE[V]=' ' THEN R:=0
           ELSE R:=(INTEGER(HEADLINE[V])-48);
        IF HEADLINE[W]=' ' THEN S:=0
           ELSE S:=(INTEGER(HEADLINE[W])-48);
        X:=P*1000+Q*100+R*10+S;
END;
{**************************** PLOTPOINT ****************************}

PROCEDURE PLOTPOINT(X,Y:INTEGER);
                                          {plots dot at coordinates x,y}
BEGIN
     SETFILLSTYLE(SOLIDFILL,PC);
     LATLONCON(X,Y);
     IF VISIBLE THEN
        BEGIN
             IF OLDVIS THEN
                BEGIN
                     MOVETO(OLDLON,OLDLAT);
                     LINETO(WINLON,WINLAT);
                END;
```

```
        FILLELLIPSE(WINLON,WINLAT,2,2);
        OLDLAT:=WINLAT;
        OLDLON:=WINLON;
        OLDVIS:=TRUE;
    END
    ELSE
    BEGIN
        OLDVIS:=FALSE;
    END;
END;

{**************************** PLOTKSC ****************************}

PROCEDURE PLOTKSC;

VAR
    NM       : INTEGER;

BEGIN
    SETFILLSTYLE(SOLIDFILL,5);
    HERELAT:=ROUND(28.3833*60);
    HERELON:=ROUND(80.5833*-60);
    LATLONCON(HERELAT,HERELON);
    FILLELLIPSE(WINLON,WINLAT,2,2);  {PLOTS KSC AS A RED DOT}
    HERELON:=WINLON;
    HERELAT:=WINLAT;

    NM:=12;                    {NM=1 DEG = 60NM}
    SETCOLOR(WHITE);                                   {PLOTS EVENT CIRCLES}
    CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*8/60));
    CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*12/60));
    CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*24/60));
    CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*36/60));
    CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*48/60));
    CIRCLE(HERELON,HERELAT,ROUND(NM*VEL*72/60));
END;
{**************************** ERASETRACK ****************************}

PROCEDURE ERASETRACK;                 {after track is plotted, replotted to}
                                      {erase it}
BEGIN
    K:=K*4;
    FOR I:=1 TO K DO
        BEGIN
            PLOTPOINT(LT[I]*6,LN[I]*-6);
        END;
        READLN(DUMMY);
        CLOSEGRAPH;
    FOR I:=1 TO 20 DO
        BEGIN
          WRITELN('LAT= ',LT[I],'  LON= ',LN[I]);
        END;
END;
{**************************** GETSTORM ****************************}
```

517

```
PROCEDURE GETSTORM;                       {gets storm data}

BEGIN                                     {selects and resets storm file}
    ASSIGN(INFILE,IN_FILE);
    RESET(INFILE);

    WHILE NOT EOF(INFILE) DO
        BEGIN
        OLDVIS:=FALSE;
        VISIBLE:=FALSE;
            FOR I:=1 TO 52 DO             {reads header line, stores data}
                BEGIN
                    READ(INFILE,HEADLINE[I]);
                END;
            READ(INFILE,C);
            HEADLINE[53]:=C;
            ENDOFLINE;

            MAKETWO(20,21);
              K:=X;
            MAKEFOUR(31,32,33,34);                {data conversion & storage}
              NMBR:=X;
            MAKETWO(7,8);
              MO:=X;
            MAKETWO(10,11);
              DY:=X;
            MAKEFOUR(13,14,15,16);
              YR:=X;
            FOR I:=1 TO 10 DO
                BEGIN
                NAME[I]:=HEADLINE[35+I]
                END;
        FOR JJ:=1 TO K DO                         {reads daily track data}
            BEGIN
            FOR I:=1 TO 81 DO
            BEGIN
                READ(INFILE,C);
                HEADLINE[I]:=C;
            END;
                READ(INFILE,C);
            MAKETWO(7,8);
              MO:=X;
            MAKETWO(10,11);
              DY:=X;
            FOR I:=1 TO 4 DO
            BEGIN
                J:=(I-1)*17;
                MAKETHREE(13+J,14+J,15+J);
                  LT[I+(JJ-1*4)]:=X;
                MAKETHREE(17+J,18+J,19+J);
                  LN[I+(JJ-1*4)]:=X;
                MAKETHREE(21+J,22+J,23+J);
                  WN[I+(JJ-1*4)]:=X;
                  IF WN[I]<35 THEN PC:=3
                  ELSE IF WN[I]<65 THEN PC:=14
```

```
                    ELSE PC:=60;
                    PLOTPOINT(LT[I+(JJ-1*4)]*6,LN[I+(JJ-1*4)]*-6);
                  MAKEFOUR(25+J,26+J,27+J,28+J);
                    SP[I+(JJ-1*4)]:=X;
              END;
            END;
            ENDOFLINE;
        END;
CLOSE(INFILE);
END;

{************************** WRITE TO SCREEN **************************}

PROCEDURE WRITE_TO_SCREEN(TOSCREEN:STRING);    {writes text to screen}

BEGIN
     OUTTEXTXY(0,400,TOSCREEN);
END;

{****************************** MAKEMAP ******************************}

PROCEDURE MAKEMAP;
{procedure to generate the map of the region. Newpoint.dat is a file of
datapoints from the Micro World Data Base. Documentation for that database
is found in the file MWDBINFO.DOC.}

BEGIN
     ASSIGN(INFILEPT,'NEWPOINT.DAT');
     RESET(INFILEPT);                     {opens and resets datafile}
   WHILE NOT EOF(INFILEPT) DO
    BEGIN
      READ(INFILEPT, NEWPOINT);           {Read datapoint, determine if it is}
      WITH NEWPOINT DO                    {a header line or a data line}
          IF CODE > 5 THEN
        BEGIN
          LATLONCON(LAT,LON);             {converts integer lat/lon to a screen}
          IF VISIBLE THEN                 {coordinate for plotting}
             BEGIN
               MOVETO(winlon,winlat);
               oldlat:=winlat;
               oldlon:=winlon;
               oldvis:=visible;
             END
          ELSE OLDVIS:=FALSE

        END
     ELSE
        BEGIN
          LATLONCON(LAT,LON);
          IF NOT VISIBLE THEN
             OLDVIS :=FALSE
          ELSE IF NOT OLDVIS THEN
               BEGIN
                    MOVETO(WINLON,WINLAT);
                    OLDLAT:=WINLAT;
                    OLDLON:=WINLON;
                    OLDVIS:=TRUE;
               END
          ELSE
               BEGIN
                    LINETO(WINLon,WINLat);
                    oldlat:=winlat;
                    oldlon:=winlon;
                    oldvis:=TRUE;
               END
           END
      END;                              {completes the outline part}
      OLDVIS:=FALSE;
    SETFILLSTYLE(SOLIDFILL,CYAN);       {makes the ocean cyan in color}
    FLOODFILL(600,100,WHITE);
    FLOODFILL(10,300,WHITE);            {fills the atlantic}
    CLOSE(INFILEPT);
    LATLONCON(610,-4300);
    FLOODFILL(WINLON,WINLAT,WHITE);     {fills pacific}
    SETFILLSTYLE(SOLIDFILL,CYAN);
    LATLONCON(1520,-4850);
    FLOODFILL(WINLON,WINLAT,WHITE);     {fills lake okechobee}
    LATLONCON(700,-5130);
    FLOODFILL(WINLON,WINLAT,WHITE);     {fills lake nicaragua}

END;

{************************** PRINTGRAPH ****************************}

PROCEDURE PRINTGRAPH;                  {prints data to epson printer}

VAR
   I,J,K,M,MSB,LSB,MAXX,MAXY: INTEGER;
BEGIN
     MAXX:= GETMAXX;
     MAXY:= GETMAXY;
     SETVIEWPORT(0,0,MAXX,MAXY,FALSE);
     WRITE(LST,ESC,'A',#$07);
          LSB:=MAXY AND $00FFF;
          MSB:=MAXY SHR 8;
          J:=0;
          REPEAT
                WRITE(LST,ESC,'*',CHAR(5),CHAR(LSB),CHAR(MSB));
                FOR I:= MAXY DOWNTO 0 DO
                BEGIN
                     M:=0;
                     FOR K:=0 TO 7 DO
                     BEGIN
                          M:= M SHL 1;
                          IF GETPIXEL(J+K,I)<> 0
                             THEN INC(M);
                     END;
                     WRITE(LST,CHAR(M));
                END;
                WRITE(LST,#$0D,#$0A);
                INC(J,8);
```

```
        UNTIL J>= MAXX-1;
    WRITE(LST,#$0C);
END;
{-.......................... MAIN PROGRAM  ............................}

BEGIN

     CLRSCR;
     WRITELN('WHAT IS THE NAME OF YOUR STORM DATA FILE?');
     READLN(IN_FILE);
     WRITELN('HOW FAST IS THE CURRENT STORM MOVING?');
     READLN(VEL);
     INITIALIZE;
     BORDER;
     SETFILLSTYLE(SOLIDFILL,GREEN);          {Fills screen with GREEN }
     FLOODFILL(100,100,WHITE);
     oldlat:=0;
     oldlon:=0;
     oldvis:=false;

          MAKEMAP;                    {draws map}
          GRIDLINES;                  {draws grids}
          BORDER;                     {makes border}
          PLOTKSC;                    {plots KSC}
          SETWRITEMODE(XORPUT);       {sets mode for writing and erasing track}
          GETSTORM;                   {draws storm}
          MoveTo(0,400);
          OUTTEXT('STORM DATA USING DATA FILE ');
          FOR I:=1 TO 11 DO OutText (IN_FILE[I]);
          READLN(DUMMY);
          {PRINTGRAPH;    }
          OUTTEXTXY(0,420,'Press return to exit.');
          CloseGraph;               {all done!}
END.   {MAIN PROGRAM}
```