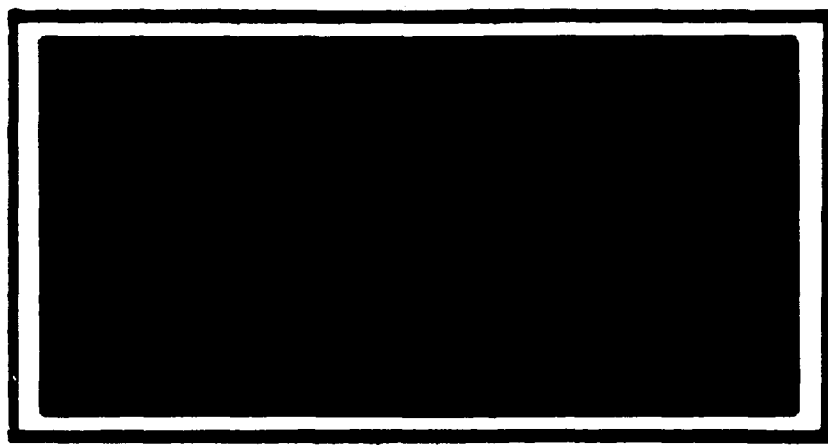


2

AD-A229 250



DTIC
ELECTE
DEC 11 1990
S D
C



DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

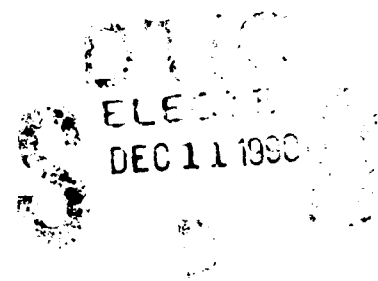
DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

90 12 10 112

2

AFIT/GLM/LSM/90S-12



AN AIR BASE VULNERABILITY ASSESSMENT
ANALYSIS TOOL FOR
U.S. AIR FORCE WAR PLANNERS
VOLUME II: TECHNICAL REFERENCE MANUAL

THESIS

Richard M. Cockley
Captain, USAF

AFIT/GLM/LSM/90S-12

Approved for public release; distribution unlimited

The opinions and conclusions in this paper are those of the author and are not intended to represent the official position of the DOD, USAF, or any other government agency.



Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Availability/ or Special
A-1	

AFIT/GLM/LSM/90S-12

AN AIR BASE VULNERABILITY ASSESSMENT ANALYSIS
TOOL FOR U.S. AIR FORCE WAR PLANNERS
VOLUME II: TECHNICAL REFERENCE MANUAL

THESIS

Presented to the Faculty of the School of Systems and
and Logistics of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Logistics Management

Richard M. Cockley
Captain, USAF

September 1990

Approved for public release; distribution unlimited

Forward

This volume contains the program documentation for the pre- and post-processor BasePlot.

Chapter I, Data Dictionary, contains a description of data in BasePlot. Chapter II, Definition Sub-Programs and Sub-Functions, contains a brief description of each individual sub-program or sub-function. Chapter III, Program Documentation, contains QuickBASIC 4.5 program code written for BasePlot.

Application and BasePlot's User's Manual are documented in Volume I: Development and User's Manual.

Table of Contents

	Page
Forward	ii
Abstract	iv
I. Data Dictionary	1
II. Definition of Sub-Programs and Sub-Functions . .	4
Sub-Programs	4
Sub-Functions	10
III. Program Documentation	11
Introduction	11
BasePlot Programing Code	11

Abstract

BasePlot's, a pre- and post-processor for TSARINA, Volume II: Technical Reference Manual contains three chapters. Chapter I, Data Dictionary, contains a description of data in BasePlot. Chapter II, Definition Sub-Programs and Sub-Functions, contains a brief description of each individual sub-program or sub-function. Chapter III, Program Documentation, contains QuickBASIC 4.5 program code written for BasePlot.

Application and BasePlot's User's Manual are documented in Volume I: Development and User's Manual.

I. Data Dictionary

Variables

A(Single Presision) = First X-coordinate
AAF(Integer) = Active attack file
AF(Integer) = Active file
AHF(Integer) = Active hit file
AV(Integer) = Active view
AW(Integer) = Active window
B(Single Precision) = First Y-coordinate
BGRD(String) = Background
BGLDCOLR(Integer) = Bold color
BOMB(Integer) = Number of bombs
C(Single Precision)= Second X-coordiante
CHAR(Integer) = Character
COL(Integer) = Colum
COLR(Integer) = Color
D(Single Precision) = Second Y-coordinate
EXT(String) = File extension
FGRD(Integer) = Foreground
FILENAME(String) = Filename
FTYPE(String) = File type
FIRSTATK(Integer) = First attack
FIRSTHIT(Integer) = First hit
FIRSTTRL(Integer) = First trial
GSTEP(Integer) = Grid step
H(Single Precision) = Length of target
INC(Integer) = Increment between bombs

ISTART(Integer) = Initial start
ISTOP(Integer) = Initial stop
LASTATK(Integer) = Last attack
LASTHIT(Integer) = Last hit
LASTTRL(Integer) = Last trial
MENUCOLR(Integer) = Menu color
MSG(String) = Message
NAF(Integer) = Number of active files
NAME(String) = Name of file(includes path and extension)
NF(Integer) = Number of files
NHF(Integer) = Number of hit files
NUM(Integer) = Number
NUMATTACKS(Integer) = Number of attacks
NUMHITS(Integer) = Number of hits
NUMTARGETS(Integer) = Number of targets
NUMTRIAL(Integer) = Number of trials
OFFSET(Integer) = Off set
OPTN(Integer) = Option
PATH(String) = Path includes the drive and any
 sub-directories
PF(Integer) = Pan Factor
PHI(Integer) = Angle off X-Y coordinates in radians
POPTN(Integer) = Pallet option
PROMPT(String) = Prompts user for inputs
ROW(Integer) = Row
SROW(Integer) = Sub-title row
STAT(Integer) = Status

TEXT(String) = Text is used to display messages

TITLE(String) = Title

TRL(Integer) = Trial

VMAX(Integer) = Maximum vertical pixels

WPNNUM(Integer) = Weapon Number

X(Integer) = X-coordinate

XMAX(Single Precision) = Maximum X-coordinate

Y(Integer) = Y-coordinate

ZF(Integer) = Zoom factor

II. Definition of Sub-Programs and Sub-Functions

Sub-Programs

Main Program	The main program initializes the variables and controls the calling of the sub-programs.
AttackControl	Shows how many active ATTACK files there are and then plots the attacks.
ChangePalette	Allows the user to change color options.
ClearAttacks	Removes attacks from the active window.
ClearControl	Determines if the user wants ATTACKS or HITS cleared from the screen and then removes them from the screen.
ClearHits	Removes the HITS from the active view window.
ClearLine	Erases a line of material based on the row # used when the SUB-PROGRAM is called.
DecodeFileName	Determines the characteristics of the filename being entered by the user.
DrawWindow	Is called to draw the active windows.
DumpBW	Draws a black and white plot of the screen on a plotter.
DumpChar	Sends characters to the plotter.
DumpColor	Prints a color representation of the screen on a plotter.
DumpControl	Determines the user's plotter characteristics.
DumpInitPrn	Sends initial codes to the plotter.
DumpInitScrn	Clears unnecessary information from the screen before printing on the plotter.
DumpLine	Sends one line of information to the plotter for printing.
DumpResetPrn	Resets printer controls.

FileErrMsg	Used if there is an error inputting a file name.
FillHitPtr	Used to color in buildings if the Target type is one that is colored in.
GetAttacks	Reads ATTACK file information into memory; based on the extension it determines if it reads old ATTACK files or new ATTACK files.
GetBounds	Draws specific points on the screen for each target.
GetHits	Processes user's HIT file input requirements.
GetTargets	Reads target data from a Target file.
GetTgtData	Reads new target colors.
GetTitle	Determines the name of the base from the user. The user can input any name but it would normally be the base being simulated.
GetWpnData	Reads in the weapon color data. If the user wishes to change the weapon color data the user would need to update the text file called DemoWpn.
HitControl	Asks the user which attack and trial the user wants shown on the screen and shows the hits for that attack and trial.
InitCoordinates	Sets up the initial coordinates for the base based on the maximum X coordinate read off the target data file.
InitPalette	Initializes pallet colors based on the DATA statement provided in the main program.
InitTargets	Initializes the initial target colors by entering integer numbers into the target color array and target fill rray from DATA statements found in the main program.
InitWeapons	Initializes the weapon colors based on the DATA statements found in the main program.
InputControl	Determines which files the users want opened based on their selection.

Intro	Brings up the initial screen with the disclaimer.
PanControl	Determines a new reference point for the program based on user inputs (left, right, down, or up).
PanCoordinates	Changes the screen reference point. The reference point is changed by moving the coordinate system on the screen.
PlotAimPair	Draws the individual circles representing the area affected by individual hits or bombs.
PlotAimPts	Determines if there is more than one bomb and calls the sub-program that draws the individual hits. The number of bombs is read from the attack cards. Each bomb stick has a certain number of bombs depending on the weapon type.
PlotAllAttacks	Is called from PlotAttack and it draws all the attack files that are active.
PlotAllHits	Is called from the PlotHits sub-program and it draws all the hits for the active files.
PlotAttacks	Is called from the Redraw window sub-program. It redraws attacks on the screen after the program updates user's requests. For example, if the user zooms into a new area of the base, the program changes the coordinates and then redraws the attacks based on the new coordinates.
PlotBorder	Defines the initial graphics areas and draws a border around the area that will represent the base.
PlotDirec	Uses the attack information to plot the direction of the bomb stick (length and width of the area affected by the bombs).
PlotGrid	Draws a grid on the screen to help locate targets and hits.
PlotGridAxis	Draws circles on the each axis of the grid to help locate the different axis numbers.
PlotGridLabels	Labels the grids based on the initial coordinates.

PlotGidLines	Draws the lines on the grid.
PlotHitControl	Determines how many hits to plot and then plots the individual hits on the screen.
PlotHits	Is called from the Redraw sub-program and is used to plot all the individual hits in the active hit file.
PlotOneAttack	Uses the attack data and plots the attack on the screen.
PlotOneHit	Plots the individual hits on the screen.
PlotStick	Determines the bomb stick starting and ending point and draws a line between the two points representing the stick.
PlotSubTitle	Shows attack and hit file information (File, attack, time of day, day of attack) on line 23.
PlotTargets	Takes the coordinates found in the TARGETS text file and draws lines to represent buildings, runways, and taxiways.
PlotTitle	Prints the title of the base being simulated plus any active attack and hit files on the top of the screen.
PrintErrMsg	Is used to print error information on line 24. It is called from the Error traps in the main program.
PrintLine	Prints a line of information based on the memory variables input from other modules. For example the test string variable might contain a question asking for a user input.
PrintMenu	Prints the main menu on the screen at row 25.
ReadNewAttacks	Reads attack text file which is in TSARINA card column format.
ReadNewHits	Reads a hit text file which is output from TSARINA.
ReadNewTargets	Reads a target text file which is in TSARINA card column format.
ReadOldAttacks	Reads files with .\$1\$ and .\$\$\$ extensions. These files are in binary format which

were created after reading the initial Attack files in TSARINA format.

ReadOldHits	Reads files with .\$. and .\$\$\$ extensions. These files are in binary format which were created after reading the initial Hit files in TSARINA format.
ReadOldTargets	Reads files with .\$. and .\$\$\$ extensions. These files are in binary format which were created after reading the initial Target files in TSARINA format.
ReDrawWindow	Used to redraw the active window whenever there are changes made to the inputs of that window.
ResetControl	Resets various controls in the main program.
ResetMatching	Determines active windows and sets original graphics coordinates within each window.
ResetSplitCoord	Resets the split coordinates to be used when using split screens.
ResetStartup	Returns the screens to the original coordinates used prior to zooming or panning.
ResetView	Resets the graphics area to its maximum size.
RestoreWindow	Restores the current active windows to graphics arrays.
SaveWindow	Saves current window to graphic arrays so they can be recalled later.
SetSplitCoord	Determines the initial split coordinates to be used whenever the user decides to view two windows on the screen.
SetWpnStat	Determines the weapon status for each weapon type.
SplitControl	Used to split the graphics area in half to allow the user to view two windows at once.
ToggleActFile	Switches the file that is currently active. There can be up to two files (Attack and Hit) open at the same time but

the user can only view one file at a time. The active files are displayed in bold white on the title line.

ToggleBGrd	Changes the color of the background. Turning background colors off allows the user to see the attacks and hits more clearly.
ToggleControl	Determines what the users wants to turn on or off by toggling certian program characteristics.
ToggleEffects	Turns on the effects for displaying attacks, hits, or the grid.
ToggleFGrd	Changes the foreground colors based on weapon status. Turning foreground colors off and then using the function keys allows the users to clearly see individual weapon types.
ToggleGrid	Turns the grid system on and off.
ToggleScreen	Changes which screen is active by changing the color of the border around the screen.
ToggleUXOs	Determines whether the unexploded ordinance is shown on screen.
ToggleWpn	Changes the colors of the weapons displayed on the screen.
WriteAttacks	Writes a binary file of the TSARINA format text file to allow for a quicker display of inputs the next time program is called.
WriteHits	Writes a binary file of the TSARINA format text file to allow for a quicker display of inputs the next time program is called.
WriteTargets	Writes a binary file of the TSARINA format text file to allow for a quicker display of inputs the next time program is called.
ZoomControl	Changes the value of the coordinate system to allow the user to get a closer view of various sections of the base.
ZoomCoordinates	Determines the new coordinate values based on whether the user wants zoom in or out.

Sub-Functions

GetFileName	Used to get filenames for Attack, Hit, Target Data, and Weapon Data files which are used as input files.
GetFileNum	Asks the user which active file number they want to remove when the number of active files exceeds the max allowed.
GetIData	Called when the user is required to tell the program which attack or trial to use when plotting hits or attacks on the screen. It performs an initial check to make sure the user is within the program parameters.
GetOptn	Used to wait for the user's responses during menu options.
GridStep	Sets the amount of space between each grid line when the grid feature is toggle on the screen.
Imax	Determines the initial maximums used by the PlotGrid sub-program.
IMin	Determines the initial minimums used by the PlotGrid sub-program.

III. Program Documentation

Introduction

TSARINA Background. BasePlot was designed to allow analysts experienced in the use of TSARINA (Theater Simulation of Air base Resources INputs using AIDA) and a knowledge of ABO planning to observe on screen the results of an attack scenario run in TSARINA. TSARINA is a Monte Carlo computer simulation model (Emerson, 1982:1) which assesses an air base's vulnerability to an enemy's conventional or chemical attack. TSARINA can be run on a main-frame or micro-computer but it requires the user to have an extensive working knowledge of ABO. TSARINA allows analysts the opportunity to simulate attacks on various airbases but it does not provide any graphical representations of either TSARINA inputs or TSARINA results.

BasePlot Programing Code

Program...BP7 (BasePlot Version 7)
Author....Capt Bob O'Neil
Editor....Capt Rick Cockley
Date.....August 1990

```
*****
REM This is the Main Module
*****

REM DECLARE indicates the number of parameters and data type
REM of each parameter that is passed using FUNCTIONS or
REM SUB-PROGRAMS

DECLARE SUB Intro ()

DECLARE SUB FileErrMsg (Num%, Msg$)

DECLARE SUB PlotGridLabels (IStart%, IStop%, GStep%, Col%,
AV%, AW%)
```

```

DECLARE SUB PlotGridAxis (IStart%, IStop%, GStep%, Colr%)
DECLARE SUB PlotGridLines (IStart%, IStop%, GStep%, Colr%,
AV%)
DECLARE FUNCTION GridStep% (A!)
DECLARE FUNCTION IMax% (A!, B!)
DECLARE FUNCTION IMin% (A!, B!)
DECLARE SUB ToggleGrid (NAF%, NHF%, AW%, AV%)
DECLARE SUB PlotGrid (AV%, AW%)
DECLARE SUB FillHitPtr (FirstAtk%, LastAtk%, FirstTrl%,
LastTrl%, AHF%, Num%)
DECLARE SUB DumpControl (BoldColr%, DefColr%, AW%, AV%)
DECLARE SUB DumpColor ( )
DECLARE SUB DumpBW ( )
DECLARE SUB DumpChar (Char%)
DECLARE SUB DumpLine (Colr%)
DECLARE SUB DumpResetPrn ( )
DECLARE SUB DumpInitPrn ( )
DECLARE SUB DumpInitScrn ( )
DECLARE SUB ResetControl (BoldColr%, DefColr%, AAF%, NAF%,
AHF%, NHF%, AV%, AW%)
DECLARE SUB ResetMatching (NAF%, NHF%, AV%, AW%)
DECLARE SUB ResetStartup (NAF%, NHF%, AV%, AW%)
DECLARE SUB ResetView (BoldColr%, DefColr%, AAF%, NAF%,
AHF%, NHF%, AV%, AW%)
DECLARE SUB ResetSplitCoord (AV%)
DECLARE SUB SetSplitCoord (AV%, AW%)
DECLARE SUB SplitControl (DefColr%, AAF%, NAF%, AHF%, NHF%,
AV%, AW%)
DECLARE SUB ToggleControl (BoldColr%, DefColr%, NAF%, AAF%,
NHF%, AHF%, AV%, AW%, BGrd$, FGrd$)

```

```

DECLARE SUB ToggleEffects (NAF%, NHF%, AW%, AV%)

DECLARE SUB ToggleScreen (AAF%, NAF%, AHF%, NHF%, AW%, AV%,
DefColr%)

DECLARE SUB DrawWindow (AAF%, NAF%, AHF%, NHF%, AW%, AV%,
Colr%)

DECLARE SUB ToggleActFile (AF%, NF%)

DECLARE SUB ToggleBGrd (BGrd$)

DECLARE SUB ToggleFGrd (FGrd$)

DECLARE SUB ToggleUXOs (NAF%, NHF%, AW%, AV%)

DECLARE SUB ChangePalette (Offset%, POptn%)

DECLARE SUB SetWpnStat (Stat%)

DECLARE SUB InitPalette ()

DECLARE SUB ClearControl (BoldColr%, DefColr%, AV%, AW%,
AAF%, NAF%, AHF%, NHF%)

DECLARE SUB ClearAttacks (NAF%, AV%)

DECLARE SUB ClearHits (NHF%, AV%)

DECLARE SUB PanControl (BoldColr%, DefColr%, PF%, AV%, AW%,
NAF%, NHF%)

DECLARE SUB ZoomControl (BoldColr%, DefColr%, ZF%, AV%, AW%,
NAF%, NHF%)

DECLARE SUB ReDrawWindow (NumTargets%, AV%, AW%, NAF%, NHF%)

DECLARE SUB ZoomCoordinates (AV%, AW%, AF%, BF%, CF%)

DECLARE SUB PanCoordinates (AV%, Optn%, PF%)

DECLARE SUB RestoreWindow (AW%, AV%)

DECLARE SUB PlotAttacks (NAF%, AV%)

DECLARE SUB PlotHits (NHF%, AV%)

DECLARE SUB HitControl (BoldColr%, DefColr%, NHF%, AHF%,
AAF%, AV%, AW%)

DECLARE SUB PlotHitControl (FirstHit%, LastHit%, FirstTrl%,
LastTrl%, AHF%, AV%)

DECLARE SUB PlotAllHits (NumHits%, NumTrials%, AHF%, AV%)

```

```

DECLARE SUB PlotOneHit (Num%, Trl%, AHF%, AV%)

DECLARE SUB AttackControl (BoldColr%, DefColr%, NAF%, AAF%,
AV%, AW%)

DECLARE SUB PlotAllAttacks (AAF%, AV%, NumAttacks%)

DECLARE SUB PlotOneAttack (Num%, AAF%, AV%)

DECLARE SUB PlotSubTitle (AV%, AW%, AAF%, AHF%)

DECLARE SUB InitCoordinates (XMax!, Y1%, Y2%)

DECLARE SUB WriteTargets (Path$, Name$, NumTargets%, XMax!)

DECLARE SUB ReadNewTargets (Path$, Name$, Ext$, NumTargets%,
XMax!)

DECLARE SUB ReadOldTargets (Path$, Name$, NumTargets%,
XMax!)

DECLARE SUB GetTargets (BoldColr%, DefColr%, NumTargets%,
XMax!)

DECLARE SUB GetTitle (Title$)

DECLARE SUB ToggleWpn (WpnNum%)

DECLARE FUNCTION GetOptn$ (Row%, Col%, Prompt$)

DECLARE SUB PrintMenu (MenuColr%, DefColr%)

DECLARE SUB InitTargets ()

DECLARE SUB InitWeapons ()

DECLARE SUB PlotBorder (AW%, AV%, Colr%)

DECLARE SUB PlotTitle (BoldColr%, DefColr%, AAF%, AHF%)

DECLARE SUB PlotTargets (NumTargets%)

DECLARE SUB SaveWindow (AW%, AV%)

DECLARE SUB ReadOldAttacks (Path$, Name$, NumAttacks%, AAF%)

DECLARE SUB GetAttacks (BoldColr%, DefColr%, NAF%, AAF%)

DECLARE FUNCTION GetFileName$ (InvalidName$, FType$)

DECLARE SUB ReadNewAttacks (Path$, Name$, Ext$, NumAttacks%,
AAF%)

DECLARE SUB DecodeFileName (FileName$, Path$, Name$, Ext$,

```

```

DECLARE SUB WriteAttacks (Path$, Name$, NumAttacks%, AAF%)

DECLARE FUNCTION GetFileNum% (BoldColr%, DefColr%, Name$(),
FType$)

DECLARE FUNCTION GetIData% (Row%, Prompt$, Min%, Max%)

DECLARE SUB WriteHits (Path$, Name$, NumHits%, NumTrials%,
AHF%)

DECLARE SUB ReadNewHits (Path$, Name$, Ext$, NumHits%,
NumTrials%, AHF%)

DECLARE SUB ReadOldHits (Path$, Name$, NumHits%, NumTrials%,
AHF%)

DECLARE SUB GetWpnData (BoldColr%, DefColr%)

DECLARE SUB GetTgtData (BoldColr%, DefColr%)

DECLARE SUB GetHits (BoldColr%, DefColr%, NHF%, AHF%)

DECLARE SUB InputControl (BoldColr%, DefColr%, AAF%, NAF%,
AHF%, NHF%)

DECLARE SUB ClrLine (Row%)

DECLARE SUB PrintLine (Row%, Col%, text$)

DECLARE SUB PrintErrMsg (Num%, Msg$)

DECLARE SUB PlotDirec (X%, Y%, W!, H!, Phi!, Colr%)

DECLARE SUB PlotStick (X%, Y%, W!, H!, Phi!, Colr%)

DECLARE SUB PlotAimPts (Bomb%, X%, Y%, Ofst!, Inc!, Phi!,
Colr%, R!, SF!)

DECLARE SUB PlotAimPair (X%, Y%, W!, H!, Phi!, Colr%, R!,
SF!)

DECLARE SUB GetBounds (I%, Colr%, XW%, YW%)

' Declare Data Structures as DYNAMIC. Allows memory to be
' freed when variables are not being used.

REM $DYNAMIC

' Increase Stack Size (Increases RAM memory for temporary
' quantity storage)

CLEAR , , 2048

' Declare Record Types

```

TYPE AttRecordType

Num AS INTEGER

Phi AS SINGLE

X AS INTEGER

Y AS INTEGER

Bomb AS INTEGER

SLen AS INTEGER

Wpn AS INTEGER

W AS SINGLE

Inc AS SINGLE

Ofst AS SINGLE

END TYPE

TYPE HitRecordType

Atk AS INTEGER

Trl AS INTEGER

X AS INTEGER

Y AS INTEGER

Wpn AS INTEGER

UXO AS INTEGER

Phi AS INTEGER

Alt AS INTEGER

END TYPE

' Declare Dimension Limits (Sets array maximums)

MaxTargets% = 1000: MaxAttacks% = 100: MaxTrials% = 25

MaxTgtTypes% = 30: MaxWpnTypes% = 10: MaxAttFiles% = 1

MaxHitFiles% = 2: MaxWindows% = 3: MaxViews% = 2

' Declare Weapon Data Structures

DIM WpnColr%(MaxWpnTypes%) ' Weapon Colors

DIM WpnStat%(MaxWpnTypes%) ' Weapon Display Status (1 = On)

DIM WpnX%(MaxWpnTypes%) ' Weapon Effects X-Dimension

DIM WpnY%(MaxWpnTypes%) ' Weapon Effects Y-Dimension

```

' Declare Target Data Structures

DIM Tgt(MaxTargets%, 9)          ' Target Type and Plot Points
DIM TgtColr%(MaxTgtTypes%)       ' Target Colors
DIM TgtFill%(MaxTgtTypes%)       ' Target Fill Option (1 = On)

' Declare Attack Data Structures

DIM AR AS AttRecordType          ' Attack Record for Random
                                ' Access

DIM AttPtr%(MaxAttacks%, MaxAttFiles%) ' Attack Pointers
                                ' for Random
                                ' Access

DIM AttStat%(MaxAttacks%, MaxAttFiles%) ' Attack Status (1
                                ' = On)

DIM AttDay%(MaxAttacks%, MaxAttFiles%) ' Day of Attack
DIM AttHour%(MaxAttacks%, MaxAttFiles%) ' Hour of Attack
DIM NumAttacks%(MaxAttFiles%)          ' Number of Attacks
DIM AttFile$(MaxAttFiles%)             ' Attack File Names

' Declare Hit Data Structures

DIM HR AS HitRecordType          ' Hit Record for Random
                                ' Access

DIM HitPtr%(MaxAttacks%, MaxTrials%, MaxHitFiles%) ' Hit
                                ' Pointers for R Access

DIM HitStat%(MaxAttacks%, MaxTrials%, MaxHitFiles%) ' Hit
                                ' Status (1 = On)

DIM NumHits%(MaxHitFiles%)        ' Number of Hits
DIM NumTrials%(MaxHitFiles%)       ' Number of Trials
DIM HitFile$(MaxHitFiles%)        ' Hit File Names

' Declare Program Control and Graphic Data Structures

DIM S1$(20000), S2$(20000)        ' Screen Maps
DIM SColr%(MaxViews%)             ' Screen Border Colors

```

```

DIM VY%(MaxWindows%, MaxViews%) ' Screen Physical
    ' Coordinates

DIM SRow%(MaxWindows%) ' Screen Subtitle Rows

DIM A(MaxViews%), B(MaxViews%) ' Screen Logical
    ' Coordinates

DIM C(MaxViews%), D(MaxViews%) ' Screen Logical
    ' Coordinates

DIM Attack$(MaxViews%) ' Screen Attack Subtitles

DIM Hit$(MaxViews%) ' Screen Hit Subtitles

DIM ECov$(MaxViews%) ' Weapon Effects Status

DIM UXOs$(MaxViews%) ' UXOs Status

DIM Grid$(MaxViews%) ' Grid Status

DIM PalColr%(3, 7) ' Palette Color Attributes

DIM G%(350, 4), ISum%(4) ' Screen dump graphics

' Declare Functions (Used to compute points and line
' distances to draw attacks, hits, and targets)

DEF FNX1 (X, W, H, Phi) = X - W * SIN(Phi) - H * COS(Phi)
    ' X: X-coordinate.

DEF FNY1 (Y, W, H, Phi) = Y - W * COS(Phi) + H * SIN(Phi)
    ' Y: Y-coordinate.

DEF FNX2 (X, W, H, Phi) = X - W * SIN(Phi) + H * COS(Phi)
    ' Phi: Angle off the X,Y coordinate.

DEF FNY2 (Y, W, H, Phi) = Y - W * COS(Phi) - H * SIN(Phi)
    ' W: Width of target.

DEF FNX3 (X, W, H, Phi) = X + W * SIN(Phi) + H * COS(Phi)
    ' H: Length of target.

DEF FNY3 (Y, W, H, Phi) = Y + W * COS(Phi) - H * SIN(Phi)

DEF FNX4 (X, W, H, Phi) = X + W * SIN(Phi) - H * COS(Phi)

DEF FNY4 (Y, W, H, Phi) = Y + W * COS(Phi) + H * SIN(Phi)

DEF FND (A, B, C, D) = (D - C) / (B - A) / BAR!

DEF FNT$(A) = RIGHT$(STR$(A), LEN(STR$(A)) - 1)

```

```

' Initialize Program Control Data Structures

WpnFile$ = ""           ' Clear Weapon File Name

TgtFile$ = "": TgtColr$ = ""   ' Clear Target File
    Names

AttFile$(1) = "": AttFile$(2) = "" ' Clear Attack File
    Names

HitFile$(1) = "": HitFile$(2) = "" ' Clear Hit File Names

BGrd$ = "ON": FGrd$ = "ON" ' Turn BGrd/FGrd Color On

ECov$(1) = "OFF": ECov$(2) = "OFF" ' Turn Weapon
    Effects Off

UXOs$(1) = "OFF": UXOs$(2) = "OFF" ' Turn UXOs Off

Grid$(1) = "OFF": Grid$(2) = "OFF" ' Turn Grid Off

Hit$(1) = "HITS:": Hit$(2) = "HITS:" ' Set Hit Subtitles

Attack$(1) = "ATTACKS:" ' Set Attack Subtitle

Attack$(2) = "ATTACKS:" ' Set Attack Subtitle

NHF% = 0: NAF% = 0 ' Number of Hit/Attack
    Files Open

AHF% = 0: AAF% = 0 ' Active Hit/Attack File

PF% = 1500: ZF% = 2500 ' Pan/Zoom Factor

AV% = 1: AW% = 1 ' Active View/Window

VY%(1, 1) = 15: VY%(1, 2) = 275 ' Window 1
    Y-Coordinates

VY%(2, 1) = 15: VY%(2, 2) = 135 ' Window 2
    Y-Coordinates

VY%(3, 1) = 155: VY%(3, 2) = 275 ' Window 3
    Y-Coordinates

SColr%(1) = 9: SColr%(2) = 14 ' Screen Colors, red
    and yellow

DefColr% = 7: BoldColr% = 15 ' Text Colors

SRow%(1) = 21: SRow%(2) = 11 ' Subtitle Rows

SRow%(3) = 21 ' Subtitle rows

```

```

VMax = 349                      ' Max Vertical Pixels (EGA)
SAR! = 47 / 64                  ' Screen Aspect Ratio (EGA)
XMax = 0                        ' Max Target X-Dimension

' Initialize Palette, Target and Weapon Data
RESTORE PaletteData ' Resets DATA statement to pallet data
                    ' prior to going to the Initial Pallet
                    ' SUB-PROGRAM.
CALL InitPalette

RESTORE TargetData ' Resets DATA statement to target data.
CALL InitTargets

RESTORE WeaponData
CALL InitWeapons

' Initialize Event Trapping
KEY OFF ' Turn function key display off

ON KEY(1) GOSUB TrapF1Key ' Assign subroutines to trap
    ' function keys
ON KEY(2) GOSUB TrapF2Key
ON KEY(3) GOSUB TrapF3Key
ON KEY(4) GOSUB TrapF4Key
ON KEY(5) GOSUB TrapF5Key
ON KEY(6) GOSUB TrapF6Key
ON KEY(7) GOSUB TrapF7Key
ON KEY(8) GOSUB TrapF8Key
ON KEY(9) GOSUB TrapF9Key
ON KEY(10) GOSUB TrapF10Key

FOR I = 1 TO 10 ' Enable function key trapping
    KEY(I) ON
NEXT I

```

```

' Initialize Screen and Colors
ON ERROR GOTO TrapNoEGA ' Set subroutine to trap no EGA
' error

SCREEN 9, , 0, 0 ' Set up EGA graphics 640 x 350 res

ON ERROR GOTO TrapErrors ' Set subroutine to trap all
' errors

PALETTE 1, 17 ' Set background blue

COLOR DefColr%, 0 ' Set default color (White on Black)

CLS ' Clears screen

' Print Intro Screen and Disclaimer

CALL Intro

COLOR DefColr%

CLS

' Read Target Data and Initialize Physical Coordinates

COLOR 15, 1

LINE (0, 0)-(639, 349), 7, B

LINE (0, 35)-(639, 35), 7, B

LOCATE 2, 27

PRINT "INITIAL DATA ENTRY SCREEN"

COLOR 7, 1

CALL GetTitle(Title$)

TgtFileName:

ON ERROR GOTO TrapErrors

CALL GetTargets(BoldColr%, DefColr%, NumTargets%, XMax)

COLOR DefColr%, 0

CALL InitCoordinates(XMax, VY%(1, 1), VY%(1, 2))

' Plot Initial Screen

```

```

CLS

CALL PlotTitle(BoldColr%, DefColr%, AAF%, AHF%)

CALL PlotBorder(AW%, AV%, SColr%(AV%))

CALL PlotTargets(NumTargets%)

CALL SaveWindow(AW%, AV%)

' Print Main Menu

RESTORE MainMenu

CALL PrintMenu(BoldColr%, DefColr%)

' Process User Selections Until User Quits

DO

    Optn$ = GetOptn$(23, 33, "WHAT NEXT? ")

    SELECT CASE Optn$

        CASE "A", "a"

            CALL AttackControl(BoldColr%, DefColr%, NAF%, AAF%,
                AV%, AW%)

        CASE "H", "h"

            CALL HitControl(BoldColr%, DefColr%, NHF%, AHF%,
                AAF%, AV%, AW%)

        CASE "C", "c"

            CALL ClearControl(BoldColr%, DefColr%, AV%, AW%,
                AAF%, NAF%, AHF%, NHF%)

        CASE "I", "i"

            InputFileNameError:

                CALL InputControl(BoldColr%, DefColr%, AAF%, NAF%,
                    AHF%, NHF%)

        CASE "Z", "z"

            CALL ZoomControl(BoldColr%, DefColr%, ZF%, AV%,
                AW%, NAF%, NHF%)

        CASE "P", "p"

```

```

        CALL PanControl(BoldColr%, DefColr%, PF%, AV%, AW%,
        NAF%, NHF%)

CASE "S", "s"
    CALL SplitControl(DefColr%, AAF%, NAF%, AHF%, NHF%,
    AV%, AW%)

CASE "T", "t"

    CALL ToggleControl(BoldColr%, DefColr%, NAF%, AAF%,
    NHF%, AHF%, AV%, AW%, BGrd$, FGrd$)

CASE "R", "r"

    CALL ResetControl(BoldColr%, DefColr%, AAF%, NAF%,
    AHF%, NHF%, AV%, AW%)

CASE "D", "d"

    CALL DumpControl(BoldColr%, DefColr%, AW%, AV%)

    RESTORE MainMenu

    CALL PrintMenu(BoldColr%, DefColr%)

CASE "Q", "q"

    EXIT DO                                'Quit

CASE ELSE

    BEEP                                ' Invalid response, try again

END SELECT

LOOP

CLOSE

END

' Event Trapping Subroutines

TrapF1Key:

    CALL ToggleWpn(1)

    RETURN

TrapF2Key:

    CALL ToggleWpn(2)

    RETURN

```

TrapF3Key:

CALL ToggleWpn(3)

RETURN

TrapF4Key:

CALL ToggleWpn(4)

RETURN

TrapF5Key:

CALL ToggleWpn(5)

RETURN

TrapF6Key:

CALL ToggleWpn(6)

RETURN

TrapF7Key:

CALL ToggleWpn(7)

RETURN

TrapF8Key:

CALL ToggleWpn(8)

RETURN

TrapF9Key:

CALL ToggleWpn(9)

RETURN

TrapF10Key:

CALL ToggleWpn(10)

RETURN

TrapNoEGA:

SCREEN 2

VY3(1, 2) = 150

```

VY%(2, 2) = 75

VY%(3, 1) = 90: VY%(3, 2) = 150

VMax = 199: SAR! = 5 / 12

RESUME NEXT

' Directs program what to do when it's interrupted by an
' error

TrapErrors:

SELECT CASE ERR

    CASE 5          ' Invalid CGA color

        RESUME NEXT

    CASE 53

        Msg$ = "File not found; Please reenter! Press any
        key to continue."

        CALL FileErrMsg(ERR, (Msg$))

        CALL ClrLine(24)

        Temp$ = "Please reenter filename:"

        CALL PrintLine(23, 20, (Temp$))

        INPUT ; Name$

        FileName$ = Name$

        CALL DecodeFileName(FileName$, Path$, Name$, Ext$)
        IF Name$ = "quit" THEN

            CLOSE

        END IF

        RESUME

    CASE 100        ' ReadNewHits

        Msg$ = "# of attacks > expected # of attacks;
        execution stopped."

        CALL PrintErrMsg(ERR, (Msg$))

        CLOSE

```

```

END

CASE 101          'ReadNewHits

    Msg$ = "# of trials > expected # of trials;
    execution stopped."

    CALL PrintErrMsg(ERR, (Msg$))

    CLOSE

END

CASE 102          'ReadNewAttacks

    Msg$ = "# of attacks > max # of attacks; execution
    stopped."

    CALL PrintErrMsg(ERR, (Msg$))

    CLOSE

END

CASE 103          'ReadNewHits

    Msg$ = "Attacks are out of sequence; execution
    stopped."

    CALL PrintErrMsg(ERR, (Msg$))

    CLOSE

END

CASE 104          'ReadNewHits

    Msg$ = "Trials are out of sequence; execution
    stopped."

    CALL PrintErrMsg(ERR, (Msg$))

    CLOSE

END

CASE ELSE

    Msg$ = "Execution stopped."

    CALL PrintErrMsg(ERR, (Msg$))

    CLOSE

```

```

        END

    END SELECT

TgtFileNameError:

    Msg$ = "File not found; Please reenter!  Press any key to
    continue."

    CALL FileErrMsg(ERR, (Msg$))

    CALL ClrLine(24)

    CALL ClrLine(12)

    CALL ClrLine(14)

    RESUME TgtFileName

' Menu, Target, and Weapon Data
MainMenu:

    DATA 11,25,10

    DATA 2,"INPUT  ATTACK  HIT  ZOOM  PAN  CLEAR  RESET
    SPLIT  TOGGLE  DUMP  QUIT"

    DATA 2,I,10,A,18,H,24,Z,30,P,36,C,;
    43,R,51,S,58,T,67,D,74,Q

InputMenu:

    DATA 5,24,13

    DATA 10,"ATTACK FILE  HIT FILE  TARGET FILE  WEAPON
    FILE  EXIT"

    DATA 10,A,24,H,35,T,49,W,64,X

ZoomMenu:

    DATA 4,24,13

    DATA 26,"IN  OUT  CHANGE ZF  EXIT"

    DATA 26,I,31,O,37,C,50,X

PanMenu:

    DATA 6,24,13

    DATA 18,"UP  DOWN  LEFT  RIGHT  CHANGE PF  EXIT"

```

DATA 18,U,23,D,30,L,37,R,45,C,58,X

ClearMenu:

DATA 4,24,13

DATA 26,"ATTACKS HITS BOTH EXIT"

DATA 26,A,36,H,43,B,51,X

ResetMenu:

DATA 4,24,13

DATA 13,"VIEW MATCH COORDINATES STARTUP COORDINATES
EXIT"

DATA 13,V,20,M,40,S,63,X

ToggleMenu:

DATA 9,24,13

DATA 4,"ATK FILE HIT FILE BGRD FGRD GRID UXCS
EFFECTS SCRN EXIT"

DATA 4,A,15,H,26,B,33,F,40,G,47,U,54,E,64,S,72,X

DumpMenu:

DATA 3,24,13

DATA 15,"DATAPRODUCTS ONLY: BLACK & WHITE COLOR
EXIT"

DATA 36,B,52,C,61,X

PaletteData:

DATA 17,2,3,4,5,6,7,57,58,59,60,61,62,63,4,4,4,4,4,4

TargetData:

DATA 7,2,7,7,5,5,6,6,2,6,1,6,0,0,0,7,7,7,7,7,;
3,1,3,6,6,1,2,7,7,7

DATA 1,1,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,;
1,0,1,0,0,0,1,0,0,0

WeaponData:

DATA 10,9,13,12,14,15 15,11,9,12

DATA 25,25,25,25,25,25,25,25,25,25

```

DATA 25,25,25,25,25,25,25,25,25,25

REM $STATIC

*****
REM This SUB-PROGRAM determines how many active ATTACK files
REM there are and then plots the attacks.
*****

SUB AttackControl (BoldColr%, DefColr%, NAF%, AAF%, AV%,
AW%)

    SHARED AttStat%(), AttDay%(), AttHour%(), NumAttacks%(),
    Attack$()

    IF NAF% > 0 THEN          ' Check for open ATTACK files

        Color BoldColr%

        CALL PrintLine(24, 29, ("Last Attack # is " +;
        STR$(NumAttacks%(AAF%))))

        COLOR DefColr%

        IF NumAttacks%(AAF%) > 1 THEN

            Num% = GetIDData%(23, ("What Attack"), 0,
            NumAttacks%(AAF%))

        ELSE

            Num% = 1

        END IF

        IF Num% = 0 THEN

            FOR I% = 1 TO NumAttacks%(AAF%) ' Determines #
            ' ATTACK files open and plots attacks based on
            ' on attack status.

                IF AttStat%(I%, AAF%) <> AV% AND AttStat%(I%,
                AAF%) <> 3 THEN

                    AttStat%(I%, AAF%) = AttStat%(I%, AAF%) + AV%

                    CALL PlotOneAttack(I%, AAF%, AV%) ' Plots
                    ' attack

                END IF

            NEXT I%

```

```

        Attack$(AV%) = "ATTACKS:  F" + FNT$(AAF%) + "/A*/*"
ELSEIF AttStat%(Num%, AAF%) <> AV% AND AttStat%(Num%,
AAF%) <> 3 THEN

    AttStat%(Num%, AAF%) = AttStat%(Num%, AAF%) + AV%

    CALL PlotOneAttack(Num%, AAF%, AV%)

    IF RIGHTS(Attack$(AV%), 2) <> "/*" THEN

        Attack$(AV%) = Attack$(AV%) + "  F" + FNT$(AAF%)
        + "/A" + FNT$(Num%)

        Attack$(AV%) = Attack$(AV%) + "/D" +
        FNT$(AttDay%(Num%, AAF%))

        Attack$(AV%) = Attack$(AV%) + "/" +
        FNT$(AttHour%(Num%, AAF%))

    END IF

END IF

END IF

CALL PlotSubTitle(AV%, AW%, AAF%, AHF%)  ' Shows
' ATTACK file info on line 23.

CALL ClrLine(24)  ' Clears the attack control
' sub-menu from line 25.

ELSE

    BEEP

END IF

END SUB

*****
REM This SUB-PROGRAM allows the user to change color
REM options.
*****

SUB ChangePalette (Offset%, POptn%)

' Offset% = 0 for BGrd, = 8 for FGrd

' POptn% = 1 for BGrd on, = 2 for FGrd on, = 3 for BGrd or
' FGrd off

    SHARED PalColr%()

    FOR I% = 1 TO 7

```

```

        PALETTE I% + Offset%, PalColr%(POptn%, I%)

    NEXT I%

END SUB

*****
REM This SUB-PROGRAM removes attacks from the active window.
*****

SUB ClearAttacks (NAF%, AV%)

    SHARED AttStat%(), NumAttacks%(), Attack$()

    Attack$(AV%) = "ATTACKS:"

    FOR J% = 1 TO NAF%

        FOR I% = 1 TO NumAttacks%(J%)

            IF AttStat%(I%, J%) = AV% OR AttStat%(I%, J%) = 3

                THEN

                    AttStat%(I%, J%) = AttStat%(I%, J%) - AV%

                . END IF

            NEXT I%

        NEXT J%

    END SUB

*****
REM This SUB-PROGRAM determines if the user wants ATTACKS or
REM HITS cleared from the screen and then removes them from
REM the screen.
*****

SUB ClearControl (BoldColr%, DefColr%, AV%, AW%, AAF%, NAF%,
    AHF%, NHF%)

    RESTORE ClearMenu

    CALL PrintMenu(BoldColr%, DefColr%)

    DO

        Optn$ = GetOptn$(23, 35, "CLEAR? ")

        SELECT CASE Optn$

            CASE "A", "a"

```

```

        CALL ClearAttacks(NAF%, AV%)
        CALL RestoreWindow(AW%, AV%)
        CALL PlotSubTitle(AV%, AW%, AAF%, AHF%)
        CALL PlotHits(NHF%, AV%)
        CALL PlotGrid(AV%, AW%)
        EXIT DO
CASE "H", "h"
        CALL ClearHits(NHF%, AV%)
        CALL RestoreWindow(AW%, AV%)
        CALL PlotSubTitle(AV%, AW%, AAF%, AHF%)
        CALL PlotAttacks(NAF%, AV%)
        CALL PlotGrid(AV%, AW%)
        EXIT DO
CASE "B", "b"
        CALL ClearAttacks(NAF%, AV%)
        CALL ClearHits(NHF%, AV%)
        CALL RestoreWindow(AW%, AV%)
        CALL PlotSubTitle(AV%, AW%, AAF%, AHF%)
        CALL PlotGrid(AV%, AW%)
        EXIT DO
CASE "X", "x"
        EXIT DO
CASE ELSE
        BEEP
END SELECT
LOOP

```

```

        CALL ClrLine(24)

END SUB

*****
REM This SUB-PROGRAM removes the HITS from the active view
REM window.
*****

SUB ClearHits (NHF%, AV%)

    SHARED HitStat%(), NumHits%(), NumTrials%(), Hit$()

    Hit$(AV%) = "HITS:"

    FOR K% = 1 TO NHF%

        FOR I% = 1 TO NumHits%(K%)

            FOR J% = 1 TO NumTrials%(K%)

                IF HitStat%(I%, J%, K%) = AV% OR HitStat%(I%,
                J%, K%) = 3 THEN

                    HitStat%(I%, J%, K%) = HitStat%(I%, J%, K%) -
                    AV%

                END IF

            NEXT J%

        NEXT I%

    NEXT K%

END SUB

*****
REM This SUB-PROGRAM erases a line of material based on the
REM row # used when the SUB-PROGRAM is called.
*****

SUB ClrLine (Row%)

    LOCATE Row%, 1

    PRINT SPACE$(79);

END SUB

```

```

*****:*****
REM This SUB-PROGRAM determines the characteristics of the
REM filename being entered by the user.
*****

```

```

SUB DecodeFileName (FileName$, Path$, Name$, Ext$)

```

```

    IF LEN(FileName$) > 0 THEN ' Determines the position of
    ' the period in a filename if there is a filename
    ' extension.

```

```

        Temp = INSTR(1, FileName$, ".")

```

```

        IF Temp > 0 THEN

```

```

            Ext$ = MID$(FileName$, Temp, 4) ' Establishes what
            ' file extension is.

```

```

            Path$ = LEFT$(FileName$, Temp - 1)

```

```

        ELSE

```

```

            Ext$ = ""

```

```

            Path$ = FileName$

```

```

        END IF

```

```

        Temp = 0

```

```

        DO

```

```

            Slash = Temp

```

```

            Temp = INSTR(Slash + 1, Path$, "\") ' Determines if
            ' there is a slash in the filename.

```

```

        LOOP UNTIL Temp = 0

```

```

        IF Slash > 0 THEN

```

```

            Name$ = MID$(Path$, Slash + 1, 8)

```

```

            Path$ = LEFT$(Path$, Slash)

```

```

        ELSE

```

```

            Temp = INSTR(1, Path$, ":") ' Determines if a colon
            ' is used in the file name and stores it as the path.

```

```

            IF Temp > 0 THEN

```

```

                Name$ = MID$(Path$, Temp + 1, 3)

```

```

        Path$ = LEFT$(Path$, Temp)

    ELSE

        Name$ = Path$ ' Establishes the filename without
        ' path.

        Path$ = ""

    END IF

END IF

ELSE

    Path$ = ""

    Name$ = ""

    Ext$ = ""

END IF

END SUB

*****
REM This SUB-PROGRAM is called to draw the active windows.
*****

SUB DrawWindow (AAF%, NAF%, AHF%, NHF%, AW%, AV%, Colr%)

    SHARED NumTargets%

    CALL PlotBorder(AW%, AV%, Colr%)

    CALL PlotSubTitle(AV%, AW%, AAF%, AHF%)

    CALL PlotTargets(NumTargets%)

    CALL SaveWindow(AW%, AV%)

    CALL PlotAttacks(NAF%, AV%)

    CALL PlotHits(NHF%, AV%)

    CALL PlotGrid(AV%, AW%)

END SUB

```

```

*****
REM This SUB-PROGRAM draws a black and white plot of the
REM screen on a plotter.
*****

```

```
SUB DumpBW
```

```
    SHARED G%(), VMax
```

```
    FOR I% = 639 TO 0 STEP -7
```

```
        IStop% = I% - 6
```

```
        IF IStop% < 0 THEN IStop% = 0
```

```
        FOR J% = 0 TO VMax STEP 1
```

```
            ISum% = 0
```

```
            ICode% = 1
```

```
            FOR K% = I% TO IStop% STEP -1
```

```
                IF POINT(K%, J%) > 0 THEN ISum% = ISum% + ICode%
```

```
                ICode% = ICode% *.2
```

```
            NEXT K%
```

```
            G%(J%, 1) = ISum%
```

```
        NEXT J%
```

```
        CALL DumpLine(1)
```

```
        LPRINT CHR$(3); CHR$(14);
```

```
    NEXT I%
```

```
END SUB
```

```

*****
REM This SUB-PROGRAM sends characters to the plotter.
*****

```

```
SUB DumpChar (Char%)
```

```
    SELECT CASE Char%
```

```
        CASE 3
```

```
            LPRINT CHR$(3); CHR$(3);
```

```
        CASE 13
```

```

        LPRINT CHR$(141);

CASE ELSE

        LPRINT CHR$(Char%);

END SELECT

END SUB

*****
REM This SUB-PROGRAM prints a color representation of the
REM screen on a plotter.
*****

SUB DumpColor

    SHARED G%(), ISum%(), VMax

    FOR I% = 639 TO 0 STEP -7

        IStop% = I% - 6

        IF IStop% < 0 THEN IStop% = 0

        FOR J% = 0 TO VMax STEP 1

            FOR K% = 1 TO 4

                ISum%(K%) = 0

            NEXT K%

            ICode% = 1

            FOR K% = I% TO IStop% STEP -1

                SELECT CASE POINT(K%, J%)

                    CASE 1 TO 8

                        ISum%(4) = ISum%(4) + ICode%

                    CASE 9, 11

                        ISum%(3) = ISum%(3) + ICode%

                    CASE 10

                        ISum%(3) = ISum%(3) + ICode%

                        ISum%(1) = ISum%(1) + ICode%

                    CASE 12

```

```

        ISum%(3) = ISum%(3) + ICode%
        ISum%(2) = ISum%(2) + ICode%
CASE 13
        ISum%(2) = ISum%(2) + ICode%
CASE 14, 15
        ISum%(2) = ISum%(2) + ICode%
        ISum%(1) = ISum%(1) + ICode%
CASE ELSE
END SELECT

        ICode% = ICode% * 2

NEXT K%

FOR K% = 1 TO 4
        G%(J%, K%) = ISum%(K%)
NEXT K%

NEXT J%

FOR J% = 4 TO 1 STEP -1
        Colr$ = "Q," + RIGHTS(STR$(J%), 1) + ",S"
        LPRINT CHR$(3); CHR$(2); CHR$(27); Colr$; CHR$(3);
        CALL DumpLine(J%)
        LPRINT CHR$(3); CHR$(10);
NEXT J%

        LPRINT CHR$(3); CHR$(14);

NEXT I%

END SUB

*****
REM This SUB-PROGRAM determines the user's plotter
REM characteristics.
*****

SUB DumpControl (BoldColr%, DefColr%, AW%, AV%)

```

```

    SHARED VY%(), A(), B(), C(), D()

    RESTORE DumpMenu

    CALL PrintMenu(BoldColr%, DefColr%)

    BEEP

    DO

        Optn$ = GetOptn$(23, 36, "DUMP? ")

        SELECT CASE Optn$

            CASE "B", "b"

                CALL DumpInitScrn

                CALL DumpInitPrn

                CALL DumpBW

                CALL DumpResetPrn

                EXIT DO

            CASE "C", "c"

                CALL DumpInitScrn

                CALL DumpInitPrn

                CALL DumpColor

                CALL DumpResetPrn

                EXIT DO

            CASE "X", "x"

                EXIT DO

            CASE ELSE

                BEEP

        END SELECT

    LOOP

    VIEW (20, VY%(AW%, 1))-(620, VY%(AW%, 2))

    WINDOW (A(AV%), B(AV%))-(C(AV%), D(AV%))

```

```

        CALL ClrLine(24)

END SUB

*****
REM This SUB-PROGRAM sends initial codes to the plotter.
*****

SUB DumpInitPrn

    WIDTH "LPT1:", 255 ' Sets the width of the plotter at
        ' 255 characters.

    FOR I% = 1 TO 9

        LPRINT

    NEXT I%

    LPRINT CHR$(27); "x,0,$";

    LPRINT CHR$(2); CHR$(29);

    LPRINT CHR$(27); "B,0,$"

    LPRINT CHR$(3);

END SUB

*****
REM This SUB-PROGRAM clears unnecessary information from the
REM screen before printing on the plotter.
*****

SUB DumpInitScrn

    SHARED VMax

    CALL ClrLine(23)

    CALL ClrLine(24)

    CALL ClrLine(25)

    VIEW (0, 0)-(639, VMax)

    WINDOW SCREEN (0, 0)-(639, VMax)

END SUB

```

```

*****
REM This SUB-PROGRAM sends one line of information to the
REM  plotter for printing.
*****

```

```

SUB DumpLine (Colr%)

```

```

    SHARED G%(), VMax

```

```

    FOR I% = 1 TO 84

```

```

        LPRINT CHR$(0);

```

```

    NEXT I%

```

```

    FOR I% = 0 TO VMax STEP 2

```

```

        CALL DumpChar(G%(I%, Colr%))

```

```

        CALL DumpChar(G%(I% + 1, Colr%))

```

```

        CALL DumpChar(G%(I% + 1, Colr%))

```

```

    NEXT I%

```

```

END SUB

```

```

*****
REM This SUB-PROGRAM resets printer controls.
*****

```

```

SUB DumpResetPrn

```

```

    LPRINT CHR$(3); CHR$(2);

```

```

    LPRINT CHR$(27); "B,8,$";

```

```

    LPRINT CHR$(27); "Q,4,$";

```

```

    LPRINT CHR$(12)

```

```

END SUB

```

```

*****
REM This SUB-PROGRAM is used if there is an error inputting
REM  a file name.
*****

```

```

SUB FileErrMsg (Num%, Msg$)

```

```

    BEEP

```

```

    CALL PrintLine(24, 1, ("ERROR # " + STR$(Num%) + ": " +
    Msg$))

```

```

DO

LOOP WHILE INKEY$ = ""

END SUB

*****
REM This SUB-PROGRAM is used to color in buildings if the
REM Target type is one that is colored in.
*****

SUB FillHitPtr (FirstAtk%, LastAtk%, FirstTrl%, LastTrl%,
AHF%, Num%)

    SHARED HitPtr%()

    FOR I% = FirstAtk% TO LastAtk%
        FOR J% = FirstTrl% TO LastTrl%
            HitPtr%(I%, J%, AHF%) = Num%
        NEXT J%
    NEXT I%

END SUB

*****
REM This SUB-PROGRAM reads ATTACK file information into REM
memory; based on the extension it determines if it
REM reads old ATTACK files or new ATTACK files.
*****

SUB GetAttacks (BoldColr%, DefColr%, NAF%, AAF%)

    SHARED AttFile$(), NumAttacks%()

    STATIC FileName$, Path$, Name$, Ext$ ' STATIC command
    ' causes these variables remain in this SUB-PROGRAM.

    SELECT CASE NAF%

        CASE 0, 1

            NAF% = NAF% + 1

            AAF% = NAF%

        CASE 2 ' If Number of Active Attack Files
        ' greater than 2 then one has to be removed.

            AAF% = GetFileNum%(BoldColr%, DefColr%, AttFile$(),
            ("ATTACK"))

```

```

        CLOSE AAF% + 3

END SELECT

FileName$ = GetFileName$((AttFile$(3 - AAF%)),
("ATTACK")) ' Gets filename from user.

CALL DecodeFileName(FileName$, Path$, Name$, Ext$)
' Determines if there is an extension and path.

COLOR BoldColr%

IF LEFT$(Ext$, 2) = ".$" THEN

    CALL ReadOldAttacks(Path$, Name$, NumAttacks%(AAF%),
AAF%) ' Reads in binary file.

ELSE ' If there is no extension reads new ATTACK
' file.

    CALL ReadNewAttacks(Path$, Name$, Ext$,
NumAttacks%(AAF%), AAF%)

    CALL WriteAttacks(Path$, Name$, NumAttacks%(AAF%),
AAF%) ' Writes new binary file.

END IF

COLOR DefColr%

AttFile$(AAF%) = Name$

END SUB

*****
REM This SUB-PROGRAM draws specific points on the screen for
REM each target.
*****

SUB GetBounds (I%, Colr%, XW%, YW%)

    SHARED Tgt()

    DIM X%(4), Y%(4)

    K = 1

    FOR J = 1 TO 7 STEP 2

        PSET (Tgt(I%, J), Tgt(I%, J + 1)), Colr%

        X%(K) = POINT(0)

```

```

        Y%(K) = POINT(1)

        K = K + 1

    NEXT J

    FOR J = 1 TO 3

        FOR K = J + 1 TO 4

            IF X%(K) > X%(J) THEN SWAP X%(K), X%(J)

            IF Y%(K) > Y%(J) THEN SWAP Y%(K), Y%(J)

        NEXT K

    NEXT J

    XW% = X%(1) - X%(4)

    YW% = Y%(1) - Y%(4)

END SUB

*****
REM This FUNCTION is used to get filenames for Attack, Hit,
REM Target Data, and Weapon Data files which are used as
REM input files.
*****

FUNCTION GetFileName$ (InvalidName$, FType$)

    SHARED FileName$, Path$, Name$, Ext$

    Temp$ = "Enter " + FType$ + " filename"

    Name$ = InvalidName$

    Temp = LEN(Name$)

    WHILE LEFT$(Name$, Temp) = InvalidName$

        CALL PrintLine(23, 20, (Temp$))

        INPUT ; Name$

        Temp = INSTR(1, Name$, ".") - 1

        IF Temp = -1 THEN Temp = LEN(Name$)

        IF LEFT$(Name$, Temp) = InvalidName$ THEN BEEP

    WEND

```

```

    FileName$ = Name$

    CALL DecodeFileName(FileName$, Path$, Name$, Ext$)

    IF LEFT$(Ext$, 2) = ".$" THEN
        OPEN "I", #5, Path$ + Name$ + ".$1$"
    ELSE
        OPEN "I", #6, Path$ + Name$ + Ext$
    END IF

    CLOSE #5

    CLOSE #6

    GetFileName$ = FileName$

END FUNCTION

*****
REM This FUNCTION asks the user which active file number
REM they want to remove when the number of active files
REM exceeds the max allowed.
*****

FUNCTION GetFileNum% (BoldColr%, DefColr%, Name$(), FType$)

    STATIC Temp$

    Temp$ = FType$ + " files:  1 - " + Name$(1) + ", 2 - " +
    Name$(2)

    Temp% = 39 - LEN(Temp$) / 2

    COLOR BoldColr%

    CALL PrintLine(24, Temp%, (Temp$))

    COLOR DefColr%

    Num% = GetIData%(23, ("Replace What File"), 1, 2)

    CALL ClrLine(24)

    GetFileNum% = Num%

END FUNCTION

```

```

*****
REM This SUB-PROGRAM processes user's HIT file input
REM requirements.
*****

SUB GetHits (BoldColr%, DefColr%, NHF%, AHF%)

    SHARED HitFile$(), NumHits%(), NumTrials%(), MaxAttacks%,
    MaxTrials%

    STATIC FileName$, Path$, Name$, Ext$

    SELECT CASE NHF%

        CASE 0, 1

            NHF% = NHF% + 1

            AHF% = NHF%

        CASE 2

            AHF% = GetFileNum%(BoldColr%, DefColr%, HitFile$(),
            ("HIT"))

            CLOSE AHF%

    END SELECT

    FileName$ = GetFileName$((HitFile$(3 - AHF%)), ("HIT"))

    CALL DecodeFileName(FileName$, Path$, Name$, Ext$)

    COLOR BoldColr%

    IF LEFT$(Ext$, 2) = ".$" THEN

        CALL ReadOldHits(Path$, Name$, NumHits%(AHF%),
        NumTrials%(AHF%), AHF%)

    ELSE

        NumHits%(AHF%) = GetIData%(24, ("How Many Attacks"),
        1, MaxAttacks%)

        NumTrials%(AHF%) = GetIData%(24, ("How Many Trials"),
        1, MaxTrials%)

        CALL ReadNewHits(Path$, Name$, Ext$, NumHits%(AHF%),
        NumTrials%(AHF%), AHF%)

        CALL WriteHits(Path$, Name$, NumHits%(AHF%),
        NumTrials%(AHF%), AHF%)
    
```

```

END IF

COLOR DefColr%

HitFile$(AHF%) = Name$

END SUB

*****
REM This FUNCTION is called when the user is required to
REM tell the program which attack or trial to use when
REM plotting hits or attacks on the screen. It performs an
REM initial check to make sure they are within
REM the program parameters.
*****

FUNCTION GetIData% (Row%, Prompt$, Min%, Max%)

    Temp% = 39 - LEN(Prompt$) / 2

    I% = Min% - 1

    WHILE I% < Min% OR I% > Max%

        CALL PrintLine(Row%, Temp%, (Prompt$))

        INPUT ; I%

        IF I% < Min% OR I% > Max% THEN BEEP

    WEND

    GetIData% = I%

END FUNCTION

*****
REM This FUNCTION is used to wait for the users responses
REM during menu options.
*****

FUNCTION GetOptn$ (Row%, Col%, Prompt$)

    CALL PrintLine(Row%, Col%, (Prompt$))

    A$ = ""

    WHILE A$ = ""

        A$ = INKEY$

    WEND

    PRINT A$;

```

```

    GetOptn$ = A$

END FUNCTION

*****
REM This SUB-PROGRAM reads target data from a Target file.
*****

SUB GetTargets (BoldColr%, DefColr%, NumTargets%, XMax)

    SHARED TgtFile$      'Makes TgtFile$ a global variable.

    STATIC FileName$, Path$, Name$, Ext$

    LOCATE 12, 20: INPUT "Enter target filename"; FileName$

    CALL DecodeFileName(FileName$, Path$, Name$, Ext$)
        ' Determines file type.

    COLOR BoldColr%

    IF LEFT$(Ext$, 2) = ".$" THEN

        CALL ReadOldTargets(Path$, Name$, NumTargets%, XMax)

    ELSE

        CALL ReadNewTargets(Path$, Name$, Ext$, NumTargets%,
            XMax)

        CALL WriteTargets(Path$, Name$, NumTargets%, XMax)

    END IF

    COLOR DefColr%

    TgtFile$ = Name$ ' Takes the Name$ variable returned from
                    ' the sub-programs and makes it equal to
                    ' TgtFile$.

END SUB

*****
REM This SUB-PROGRAM reads new target colors.
*****

SUB GetTgtData (BoldColr%, DefColr%)

    SHARED TgtColr$, TgtColr%(), TgtFill%(), MaxTgtTypes$

    STATIC Card$, FileName$, Path$, Name$, Ext$

    ON ERROR GOTO FileNameError

```

```

FileName$ = GetFileName$(""), ("Target Color"))
CALL DecodeFileName(FileName$, Path$, Name$, Ext$)
COLOR BoldColr%
CALL PrintLine(24, 25, ("Reading Target Color:"))
OPEN "I", #3, Path$ + Name$ + Ext$
I% = 0
WHILE NOT EOF(3) AND I% < MaxTgtTypes%
    LINE INPUT #3, Card$
    I% = I% + 1
    TgtColr%(I%) = VAL(MID$(Card$, 4, 2))
    TgtFill%(I%) = VAL(MID$(Card$, 7, 1))
    LOCATE 24, 46: PRINT I%;
WEND
CLOSE #3
COLOR DefColr%
TgtColr$ = Name$
END SUB

*****
REM This SUB-PROGRAM determines the name of the base from
REM the user. The user can input any name but it
REM would normally be the base being simulated.
*****

SUB GetTitle (Title$)
    LOCATE 8, 20: INPUT "Enter name of base"; Title$
    IF LEN(Title$) > 20 THEN Title$ = LEFT$(Title$, 20)
END SUB

```

```

*****
REM This SUB-PROGRAM reads in the weapon color data.  If the
REM user wishes to change the weapon color data the user
REM needs to update the text file called DemoWpn.
*****

```

```

SUB GetWpnData (BoldColr%, DefColr%)

```

```

    SHARED WpnFile$, WpnColr%(), WpnStat%(), WpnX%(),
    WpnY%(), MaxWpnTypes%

```

```

    STATIC Card$, FileName$, Path$, Name$, Ext$

```

```

    FileName$ = GetFileName$(""), ("Weapon"))

```

```

    CALL DecodeFileName(FileName$, Path$, Name$, Ext$)

```

```

    COLOR BoldColr%

```

```

    CALL PrintLine(24, 25, ("Reading Weapon Number:"))

```

```

    OPEN "I", #3, Path$ + Name$ + Ext$

```

```

    I% = 0

```

```

    WHILE NOT EOF(3) AND I% < MaxWpnTypes%

```

```

        LINE INPUT #3, Card$

```

```

        I% = I% + 1

```

```

        WpnColr%(I%) = VAL(MID$(Card$, 4, 2))

```

```

        WpnX%(I%) = VAL(MID$(Card$, 7, 4))

```

```

        WpnY%(I%) = VAL(MID$(Card$, 12, 4))

```

```

        LOCATE 24, 47: PRINT I%;

```

```

    WEND

```

```

    CLOSE #3

```

```

    COLOR DefColr%

```

```

    WpnFile$ = Name$

```

```

END SUB

```

```

*****
REM This FUNCTION sets the amount of space between each grid
REM line when the grid feature is toggle on the screen.
*****

```

```

FUNCTION GridStep% (A)

```

```

SELECT CASE A

```

```

    CASE IS <= 150

```

```

        GridStep% = 10

```

```

    CASE IS <= 500

```

```

        GridStep% = 50

```

```

    CASE IS <= 1500

```

```

        GridStep% = 100

```

```

    CASE IS <= 5000

```

```

        GridStep% = 500

```

```

    CASE IS <= 15000

```

```

        GridStep% = 1000

```

```

    CASE ELSE

```

```

        GridStep% = 5000

```

```

END SELECT

```

```

END FUNCTION

```

```

*****
REM This SUB-PROGRAM asks the user which attack and trial
REM the user wants shown on the screen and shows the hits
REM for that attack and trial.
*****

```

```

SUB HitControl (BoldColr%, DefColr%, NHF%, AHF%, AAF%, AV%,
AW%)

```

```

    SHARED HitStat%(), NumHits%(), NumTrials%(), Hit$()

```

```

    STATIC Trls$, Hits$

```

```

    IF NHF% > 0 THEN

```

```

        Trls$ = STR$(NumHits%(AHF%))

```

```

Hits$ = STR$(NumTrials%(AHF%))

COLOR BoldColr%

CALL PrintLine(24, 25, ("Attacks: " + Trls$ + "
Trials: " + Hits$))

COLOR DefColr%

    IF NumHits%(AHF%) > 1 THEN

        Num% = GetIData%(23, ("What Attack"), 0,
        NumHits%(AHF%))

    ELSE

        Num% = 1

    END IF

    IF NumTrials%(AHF%) > 1 THEN

        Trl% = GetIData%(23, ("What Trial"), 0,
        NumTrials%(AHF%))

    ELSE

        Trl% = 1

    END IF

    IF Num% = 0 AND Trl% = 0 THEN

        CALL PlotHitControl(1, NumHits%(AHF%), 1,
        NumTrials%(AHF%), AHF%, AV%)

        Hit$(AV%) = "HITS:  F" + FNT$(AHF%) + "/A*/T*/*"

    ELSEIF Num% = 0 THEN

        CALL PlotHitControl(1, NumHits%(AHF%), Trl%, Trl%,
        AHF%, AV%)

        Hit$(AV%) = "HITS:  F" + FNT$(AHF%) + "/A*/T" +
        FNT$(Trl%) + "/*"

    ELSEIF Trl% = 0 THEN

        CALL PlotHitControl(Num%, Num%, 1,
        NumTrials%(AHF%), AHF%, AV%)

        Hit$(AV%) = "HITS:  F" + FNT$(AHF%) + "/A" +
        FNT$(Num%) + "/T*/*"

```

```

ELSEIF HitStat%(Num%, Trl%, AHF%) <> AV% AND
HitStat%(Num%, Trl%, AHF%) <> 3 THEN

    HitStat%(Num%, Trl%, AHF%) = HitStat%(Num%, Trl%,
    AHF%) + AV%

    CALL PlotOneHit(Num%, Trl%, AHF%, AV%)

    IF RIGHT$(Hit$(AV%), 2) <> "/" THEN
        Hit$(AV%) = Hit$(AV%) + " F" + FNT$(AHF%) +
        "/A" + FNT$(Num%)

        Hit$(AV%) = Hit$(AV%) + "/T" + FNT$(Trl%)

    END IF

END IF

CALL PlotSubTitle(AV%, AW%, AAF%, AHF%)

CALL ClrLine(24)

ELSE

    BEEP

END IF

END SUB

*****
REM This FUNCTION determines the initial maximums used by
REM the PlotGrid sub-program.
*****

FUNCTION IMax% (A, B)

    IF A > B THEN

        IMax% = A

    ELSE

        IMax% = B

    END IF

END FUNCTION

```

```

*****
REM This FUNCTION determines the initial minimums used by
REM the PlotGrid sub-program.
*****

FUNCTION IMin% (A, B)

    IF A < B THEN

        IMin% = A

    ELSE

        IMin% = B

    END IF

END FUNCTION

*****
REM This SUB-PROGRAM sets up the initial coordinates for the
REM base based on the maximum X coordinate read off the
REM target data file.
*****

SUB InitCoordinates (XMax, Y1%, Y2%)

    SHARED A(), B(), C(), D()

    FOR I% = 1 TO 2

        A(I%) = 0

        B(I%) = 0

        C(I%) = XMax

        D(I%) = C(I%) * FND(20, 620, Y1%, Y2%)

    NEXT I%

END SUB

*****
REM This SUB-PROGRAM initializes pallet colors based on the
REM DATA statement provided in the main program.
*****

SUB InitPalette

    SHARED PalColr%()

    FOR I% = 1 TO 3

```

```

        FOR J% = 1 TO 7

            READ PalColr%(I%, J%)

        NEXT J%

    NEXT I%

END SUB

*****
REM This SUB-PROGRAM initializes the initial target colors
REM by entering integer numbers into the target color array
REM and target fill array from DATA statements found in the
REM main program.
*****

SUB InitTargets

    SHARED TgtColr%(), TgtFill%(), MaxTgtTypes%

    FOR I% = 1 TO MaxTgtTypes%

        READ TgtColr%(I%)

    NEXT I%

    FOR I% = 1 TO MaxTgtTypes%

        READ TgtFill%(I%)

    NEXT I%

END SUB

*****
REM This SUB-PROGRAM initializes the weapon colors based on
REM the DATA statements found in the main program.
*****

SUB InitWeapons

    SHARED WpnColr%(), WpnStat%(), WpnX%(), WpnY%(),
    MaxWpnTypes%

    FOR I% = 1 TO MaxWpnTypes%

        READ WpnColr%(I%)

        WpnStat%(I%) = 1

    NEXT I%

    FOR I% = 1 TO MaxWpnTypes%

```

```

        READ WpnX%(I%)

    NEXT I%

    FOR I% = 1 TO MaxWpnTypes%

        READ WpnY%(I%)

    NEXT I%

END SUB

*****
REM This SUB-PROGRAM determines which files the users want
REM opened based on their selection.
*****

SUB InputControl (BoldColr%, DefColr%, AAF%, NAF%, AHF%,
NHF%)

    SHARED MaxAttacks%, MaxTrials%, MaxTgtTypes%,
    MaxWpnTypes%

    RESTORE InputMenu

    CALL PrintMenu(BoldColr%, DefColr%)

    DO

        Optn$ = GetOptn$(23, 35, "INPUT? ") ' Asks user for
        ' what type of input

        SELECT CASE Optn$

            CASE "A", "a"

                CALL GetAttacks(BoldColr%, DefColr%, NAF%, AAF%)
                ' Inputs attack file.

                CALL PlotTitle(BoldColr%, DefColr%, AAF%, AHF%)
                ' Lists attack file in title.

                EXIT DO

            CASE "H", "h"

                CALL GetHits(BoldColr%, DefColr%, NHF%, AHF%)
                ' Inputs hit files.

                CALL PlotTitle(BoldColr%, DefColr%, AAF%, AHF%)

                EXIT DO

```

```

CASE "T", "t"

    CALL GetTgtData(BoldColr%, DefColr%) ' Inputs
    ' target color data.

    EXIT DO

CASE "W", "w"

    CALL GetWpnData(BoldColr%, DefColr%) ' Inputs
    ' weapon color data.

    CALL PlotTitle(BoldColr%, DefColr%, AAF%, AHF%)
    EXIT DO

CASE "X", "x"      ' Exits input option.

    EXIT DO

CASE ELSE

    BEEP

END SELECT

LOOP

CALL ClrLine(24)      ' Clears line 24 with sub-menu.

END SUB

*****
REM This SUB-Program brings up the initial screen with the
REM disclaimer
*****

SUB Intro

    SCREEN 9, , 0, 1      ' draw on invisable screen

    COLOR 7, 1

    LINE (0, 0)-(639, 349), 7, B ' draw box

    BS = "C7 BM45,120"

    ' Draw Box

    DRAW BS + "U90 R45 M+15,+15 D20 M-10,+10 M+10,+10 D20
    M-15,+15 L45"

    DRAW "BM+10,-10 U30 R30 M+10,+10 D10 M-10,+10 L30"

    DRAW "BU40 U30 R30 M+10,+10 D10 M-10,+10 L30"

```

DRAW "BR5 U26 R29 BM-29,+66 U26 R29"

DRAW "BM-44,+36 M+4,+4 R45 M+16,-16 U23 M-10,-10
M+10,-10 U20 M-15,-15 L5"

DRAW B\$ + "BM+5,-5 P14,7 BM+6,-74 P3,7 BD40 P3,7
BM+40,-6 P3,7"

'DRAW "A"

DRAW B\$ + "BR70 U65 M+10,-10 R40 M+10,+10 D65 L10 U40
L40 D40 L10"

DRAW "BM+10,-50 U10 M+5,-5 R30 M+5,+5 D10 L40 BR5 U8
M+3,-3 R31"

DRAW "BM-49,+61 M+4,+4 R11 U40 R35 BD36 M+4,+4 R11 U69
M-10,-10 L5"

DRAW "BM-40,+30 P15,7 BM-5,+47 P3,7 BR55 P3,7 BM-48,-57
P3,7"

'DRAW "S"

DRAW B\$ + "BR140 BU20 D10 M+10,+10 R40 M+10,-10 U30
M-10,-10 L35 M-5,-5 U5"

DRAW "M+5,-5 R30 M+5,+5 R10 U5 M-10,-10 L40 M-10,+10 D15
M+10,+10 R35"

DRAW "M+5,+5 D20 M-5,+5 L30 M-5,-5 U5 L10 BR10 R5 D5
M+5,+5"

DRAW "BU40 M-5,-5 U3 M+3,-3 R31 BM+1,+1 M+4,+4 R10
M+1,-1 U8 M-10,-10 L5"

DRAW "BM-40,+75 M+4,+4 R40 M+11,-11 U33 M-10,-10 L5
BM-40,+10 M+4,+4 R34"

DRAW "BM-43,+21 P15,7 BM+10,+17 P3,7 BU39 P3,7 BM-2,+22
P3,7 BU42 P3,7 BR50 P3,7"

'DRAW "E"

DRAW B\$ + "BR210 U75 R55 M+5,+5 D5 L50 D15 R35 M+5,+5
M-5,+5 L35 D30 R50"

DRAW "D5 M-5,+5 L55 M+4,+4 R55 M+6,-6 U5 M-3,-3 L2"

DRAW "BL45 U26 R33 M+8,-8 U1 M-5,-5 L5 BL31 U11 R50 U9
M-5,-5 L5"

DRAW "BM-50,+70 P15,7 BD7 P3,7 BM+8,-17 P3,7 BU40 P3,7"

'DRAW "P"

DRAW BS + "BR280 U90 R45 M+15,+15 D20 M-15,+15 L35 D40
L10"

DRAW "BM+10,-50 U30 R30 M+10,+10 D10 M-10,+10 L30 BR5
U26 R29"

DRAW "BM-44,+76 M+4,+4 R11 U40 R33 M+18,-18 U21 M-15,-15
L5"

DRAW "BM-41,+85 P14,7 BD7 P3,7 BM+8,-57 P3,7"

'DRAW "L"

DRAW BS + "BR350 U75 R10 D65 R50 D5 M-5,+5 L55 M+4,+4
R55 M+6,-6 U5 M-3,-3"

DRAW "L2 BL45 U61 M-4,-4 L1 BM-5,+70 P15,7 BD7 P3,7
BM+7,-57 P3,7"

'DRAW "O"

DRAW BS + "BR430 M-10,-10 U55 M+10,-10 R40 M+10,+10 D55
M-10,+10 L40"

DRAW "BU15 U45 M+5,-5 R30 M+5,+5 D45 M-5,+5 L30 M-5,-5"

DRAW "BD15 M+4,+4 R38 M+13,-13 U56 M-10,-10 L5"

DRAW "BM-30,+65 M-5,-5 U43 M+3,-3 R31"

DRAW "BM-44,+41 P15,7 BR8 P3,7 BM+2,+22 P3,7"

'DRAW "T"

DRAW BS + "BR515 U65 L25 U10 R60 D10 L25 D65 L10 M+4,+4
R11 U65 R25 U11"

DRAW "M-3,-3 L2 BM-60,+10 M+4,+4 R21"

DRAW "BM+5,+56 P15,7 BD7 P3,7 BM-25,-64 P3,7"

'PRINT PROGRAMER AND EDITOR

COLOR 15

LOCATE 12, 17

PRINT "Written by: Capt Bob O'Neil, Autovon 227-6520";

COLOR 7

LOCATE 14, 23

```

PRINT "Mobility and Operability Division";

LOCATE 15, 21

PRINT "Directorate for Theater Force Analysis";

LOCATE 16, 20

PRINT "Air Force Center for Studies & Analysis";

LOCATE 17, 16
PRINT "(Edited & Documented by: Capt Cockley,
  AFIT/LSG)";

' Print Disclaimer

  LOCATE 19, 5

  PRINT "This program is the property of AFSCA/SAGO;
    permission is granted to the";

  LOCATE 20, 5

  PRINT "user to make copies and distribute this program
    as long as this notice is";

  LOCATE 21, 5

  PRINT "included. While the author believes the program
    is accurate and reliable,";

  LOCATE 22, 5

  PRINT "the user assumes sole responsibility when using
    it.";

  COLOR 15

  LOCATE 24, 25

  PRINT "PRESS ANY KEY TO CONTINUE ...";

  SCREEN 9, , 0, 0

  A$ = ""

  WHILE A$ = ""

    A$ = INKEY$

  WEND

END SUB

```

```

*****
REM This SUB-PROGRAM determines a new reference point for
REM the program based on user inputs. The reference
REM point is changed by moving the coordinate system on the
REM screen.
*****

```

```

SUB PanControl (BoldColr%, DefColr%, PF%, AV%, AW%, NAF%,
NHF%)

```

```

    SHARED NumTargets%

```

```

    STATIC Temp$

```

```

    RESTORE PanMenu

```

```

    CALL PrintMenu(BoldColr%, DefColr%)

```

```

    DO

```

```

        Optn$ = GetOptn$(23, 36, "PAN? ")
        SELECT CASE Optn$

```

```

            CASE "U", "u"                                'Pans up.

```

```

                CALL PanCoordinates(AV%, 2, PF%)

```

```

                CALL ReDrawWindow(NumTargets%, AV%, AW%, NAF%,
                NHF%)

```

```

                EXIT DO

```

```

            CASE "D", "d"                                'Pans down.

```

```

                CALL PanCoordinates(AV%, 2, (-PF%))

```

```

                CALL ReDrawWindow(NumTargets%, AV%, AW%, NAF%,
                NHF%)

```

```

                EXIT DO

```

```

            CASE "L", "l"                                'Pans left.

```

```

                CALL PanCoordinates(AV%, 1, (-PF%))

```

```

                CALL ReDrawWindow(NumTargets%, AV%, AW%, NAF%,
                NHF%)

```

```

                EXIT DO

```

```

            CASE "R", "r"                                'Pans right.

```

```

                CALL PanCoordinates(AV%, 1, PF%)

```

```

        CALL ReDrawWindow(NumTargets%, AV%, AW%, NAF%,
            NHF%)

        EXIT DO

        CASE "C", "c" 'changes the pan factor to allow the
            ' user to move in bigger increments.

            Temp$ = "Old Pan Factor =" + STR$(PF%) + " New
            Pan Factor = "

            PF% = GetIData%(23, (Temp$), 0, 10000)

        CASE "X", "x"

            EXIT DO

        CASE ELSE

            BEEP

        END SELECT

    LOOP

    CALL ClrLine(24)

END SUB

*****
REM This SUB-PROGRAM changes the screen reference point.
*****

SUB PanCoordinates (AV%, Optn%, PF%)

    SHARED A(), B(), C(), D()

    SELECT CASE Optn%

        CASE 1          ' Controls left and right movement.

            A(AV%) = A(AV%) + PF%

            C(AV%) = C(AV%) + PF%

        CASE 2          ' Controls up and down movement.

            B(AV%) = B(AV%) + PF%

            D(AV%) = D(AV%) + PF%

    END SELECT

END SUB

```

```

*****
REM This SUB-PROGRAM draws the individual circles
REM representing the area affected by individual hits or
REM bombs.
*****

```

```

SUB PlotAimPair (X%, Y%, W, H, Phi, Colr%, R, SF)

```

```

    X1 = FNX1(X%, W, H, Phi)

```

```

    Y1 = FNY1(Y%, W, H, Phi)

```

```

    CIRCLE (X1, Y1), R, Colr%, , , SF

```

```

    X3 = FNX3(X%, W, H, Phi)

```

```

    Y3 = FNY3(Y%, W, H, Phi)

```

```

    CIRCLE (X3, Y3), R, Colr%, , , SF

```

```

END SUB

```

```

*****
REM This SUB-PROGRAM determines if there is more than one
REM bomb and calls the sub-program that draws the
REM individual hits. The number of bombs is read from the
REM attack cards. Each bomb stick has a certain number
REM of bombs depending on the weapon type.
*****

```

```

SUB PlotAimPts (Bomb%, X%, Y%, Ofst, Inc, Phi, Colr%, R, SF)

```

```

    IF Bomb% > 1 THEN

```

```

        CALL PlotAimPair(X%, Y%, Ofst, 0, Phi, Colr%, R, SF)

```

```

        FOR K = 2 TO Bomb% / 2

```

```

            W = Ofst + Inc * (K - 1)

```

```

            CALL PlotAimPair(X%, Y%, W, 0, Phi, Colr%, R, SF)

```

```

        NEXT K

```

```

    ELSE

```

```

        CIRCLE (X%, Y%), R, Colr%, , , SF

```

```

    END IF

```

```

END SUB

```

```

*****
REM This SUB-PROGRAM is called from PlotAttack and it draws
REM all the attack files that are active.
*****

```

```

SUB PlotAllAttacks (AAF%, AV%, NumAttacks%)

```

```

    SHARED AttStat%()

```

```

    FOR I% = 1 TO NumAttacks%

```

```

        IF AttStat%(I%, AAF%) = AV% OR AttStat%(I%, AAF%) = 3
        THEN

```

```

            CALL PlotOneAttack(I%, AAF%, AV%)

```

```

        END IF

```

```

    NEXT I%

```

```

END SUB

```

```

*****
REM This SUB-PROGRAM is called from the PlotHits sub-program
REM and it draws all the hits for the active files.
*****

```

```

SUB PlotAllHits (NumHits%, NumTrials%, AHF%, AV%)

```

```

    SHARED HitStat%()

```

```

    FOR I% = 1 TO NumHits%

```

```

        FOR J% = 1 TO NumTrials%

```

```

            IF HitStat%(I%, J%, AHF%) = AV% OR HitStat%(I%, J%,
            AHF%) = 3 THEN

```

```

                CALL PlotOneHit(I%, J%, AHF%, AV%)

```

```

            END IF

```

```

        NEXT J%

```

```

    NEXT I%

```

```

END SUB

```

```

*****
REM This SUB-PROGRAM called from the Redraw window sub-
REM program. It redraws attacks on the screen after the
REM program updates user's requests. For example, if the
REM user zooms into a new area of the base, the program
REM changes the coordinates and then redraws the attacks
REM based on the new coordinates.
*****

```

```

SUB PlotAttacks (NAF%, AV%)

```

```

    SHARED NumAttacks%()

```

```

    FOR I% = 1 TO NAF%

```

```

        CALL PlotAllAttacks(I%, AV%, NumAttacks%(I%))

```

```

    NEXT I%

```

```

END SUB

```

```

*****
REM This SUB-PROGRAM defines the initial graphics areas and
REM draws a border around the area that will represent the
REM base.
*****

```

```

SUB PlotBorder (AW%, AV%, Colr%)

```

```

    SHARED A(), B(), C(), D(), VY%()

```

```

    Y1% = VY%(AW%, 1) - 1

```

```

    Y2% = VY%(AW%, 2) + 1

```

```

    VIEW (19, Y1%)-(621, Y2%)

```

```

    WINDOW SCREEN (19, Y1%)-(621, Y2%)

```

```

    LINE (19, Y1%)-(621, Y2%), Colr%, B

```

```

    VIEW (20, VY%(AW%, 1))-(620, VY%(AW%, 2))

```

```

    WINDOW (A(AV%), B(AV%))-(C(AV%), D(AV%))

```

```

END SUB

```

```

*****
REM This SUB-PROGRAM uses the attack information to plot the
REM direction of the bomb stick (length and width of the
REM area affected by the bombs).
*****

```

```

SUB PlotDirec (X%, Y%, W, H, Phi, Colr%)

```

```

X4 = FNX4(X%, W, H, Phi)
Y4 = FNY4(Y%, W, H, Phi)
X3 = FNX3(X%, W, H, Phi)
Y3 = FNY3(Y%, W, H, Phi)

LINE (X3, Y3)-(X4, Y4), Colr%

END SUB

*****
REM This SUB-PROGRAM draws a grid on the screen to help
REM locate targets and hits.
*****

SUB PlotGrid (AV%, AW%)

  SHARED A(), B(), C(), D(), Grid$()

  IF Grid$(AV%) = "ON" THEN

    Colr% = 12

    GStep% = GridStep%((C(AV%) - A(AV%)))

    IStart% = IMin%(A(AV%), B(AV%)) \ GStep%

    IStop% = IMax%(C(AV%), D(AV%)) \ GStep% + 1

    CALL PlotGridLines(IStart%, IStop%, GStep%, Colr%,
      AV%)

    CALL PlotGridAxis(IStart%, IStop%, GStep%, 15)

    CALL PlotGridLabels(IStart%, IStop%, GStep%, Colr%,
      AV%, AW%)

  END IF

END SUB

*****
REM This SUB-PROGRAM draws circles on the each axis of the
REM grid.
*****

SUB PlotGridAxis (IStart%, IStop%, GStep%, Colr%)

  FOR I% = IStart% TO IStop%

    Temp% = I% * GStep%

```

```

        CIRCLE (Temp%, 0), 25, Colr%
        PAINT (Temp%, 0), Colr%
        CIRCLE (0, Temp%), 25, Colr%
        PAINT (0, Temp%), Colr%

    NEXT I%

END SUB

*****
REM This SUB-PROGRAM labels the grids based on the initial
REM coordinates.
*****

SUB PlotGridLabels (IStart%, IStop%, GStep%, Colr%, AV%,
AW%)

    SHARED VY%(), A(), D(), DefColr%

    PSET (A(AV%), D(AV%)), 0

    IF POINT(0) = 0 THEN

        ColAdj% = 0
        RowAdj% = 2

    ELSE

        ColAdj% = -2
        RowAdj% = 1

    END IF

    TopRow% = (VY%(AW%, 1) + 6) \ 14
    BotRow% = (VY%(AW%, 2) + 6) \ 14

    COLOR Colr%

    FOR I% = IStart% TO IStop% STEP 2

        Temp% = I% * GStep%

        PSET (Temp%, D(AV%))

        Col% = (POINT(0) + 4) \ 8 + ColAdj%

        IF Col% > 10 AND Col% < 72 THEN

```

```

        LOCATE TopRow% + 2, Col%
        PRINT Temp%;
    END IF

    PSET (A(AV%), Temp%)

    Row% = (POINT(1) + 7) \ 14 + RowAdj%

    IF Row% > TopRow% + 2 AND Row% < BotRow% THEN
        LOCATE Row%, 4
        PRINT Temp%;
    END IF

NEXT I%

COLOR DefColr%

END SUB

*****
REM This SUB-PROGRAM draws the lines on the grid.
*****

SUB PlotGridLines (IStart%, IStop%, GStep%, Colr%, AV%)

    SHARED A(), B(), C(), D()

    FOR I% = IStart% TO IStop%

        Temp% = I% * GStep%

        LINE (Temp%, B(AV%))-(Temp%, D(AV%)), Colr%

        LINE (A(AV%), Temp%)-(C(AV%), Temp%), Colr%

    NEXT I%

END SUB

*****
REM This SUB-PROGRAM determines how many hits to plot and
REM then plots the individual hits on the screen.
*****

SUB PlotHitControl (FirstHit%, LastHit%, FirstTrl%,
LastTrl%, AHF%, AV%)

    SHARED HitStat%()

```

```

FOR I% = FirstHit% TO LastHit%
  FOR J% = FirstTrl% TO LastTrl%
    IF HitStat%(I%, J%, AHF%) <> AV% AND HitStat%(I%,
      J%, AHF%) <> 3;
      THEN
        HitStat%(I%, J%, AHF%) = HitStat%(I%, J%, AHF%)
          + AV%
        CALL PlotOneHit(I%, J%, AHF%, AV%)
      END IF
    NEXT J%
  NEXT I%
END SUB

*****
REM This SUB-PROGRAM is called from the Redraw sub-program
REM and is used to plot all the individual hits in the
REM active hit file.
*****

SUB PlotHits (NHF%, AV%)
  SHARED NumHits%(), NumTrials%()
  FOR K% = 1 TO NHF%
    CALL PlotAllHits(NumHits%(K%), NumTrials%(K%), K%,
      AV%)
  NEXT K%
END SUB

*****
REM This SUB-PROGRAM uses the attack data and plots the
REM attack on the screen.
*****

SUB PlotOneAttack (Num%, AAF%, AV%)
  SHARED AttPtr%(), WpnColr%(), WpnX%(), WpnY%(), SAR!,
    ECov$()
  DIM AR AS AttRecordType
  Ptr% = AttPtr%(Num%, AAF%)

```

```

GET AAF% + 3, Ptr%, AR

WHILE AR.Num = Num%

    Colr% = WpnColr%(AR.Wpn)

    CALL PlotStick(AR.X, AR.Y, AR.W, 0, AR.Phi,
        Colr%) ' Draws bomb sticks.

    CALL PlotDirec(AR.X, AR.Y, AR.W, 50, AR.Phi,
        Colr%) ' Plots the direction of the stick.

    CALL PlotAimPts(AR.Bomb, AR.X, AR.Y, AR.Ofst, AR.Inc,
        AR.Phi, Colr%, 25, SAR!) ' Draws the individual bombs
        ' in the stick.

    IF ECov$(AV%) = "ON" AND WpnX%(AR.Wpn) > 25 THEN

        R = WpnX%(AR.Wpn)

        SF = SAR! * WpnY%(AR.Wpn) / R

        CALL PlotAimPts(AR.Bomb, AR.X, AR.Y, AR.Ofst,
            AR.Inc, AR.Phi, Colr%, R, SF)

    END IF

    Ptr% = Ptr% + 1

    GET AAF% + 3, Ptr%, AR

WEND

END SUB

*****
REM This SUB-PROGRAM plots the individual hits on the
REM screen.
*****

SUB PlotOneHit (Num%, Trl%, AHF%, AV%)

    SHARED HitPtr%(), WpnColr%(), WpnX%(), WpnY%(), SAR!,
        ECov$(), UXOs$()

    DIM HR AS HitRecordType

    HPtr% = HitPtr%(Num%, Trl%, AHF%)

    GET AHF%, HPtr%, HR

    WHILE HR.Atk = Num% AND HR.Trl = Trl%

        Colr% = WpnColr%(HR.Wpn)

```

```

IF HR.UXO = 0 THEN
    CIRCLE (HR.X, HR.Y), 25, Colr%, , , SAR!
    PAINT (HR.X, HR.Y), Colr%, Colr%
    IF ECov$(AV%) = "ON" AND WpnX%(HR.Wpn) > 25 THEN
        SF = WpnY%(HR.Wpn) / WpnX%(HR.Wpn)
        IF SF > 1 THEN
            R = WpnX%(HR.Wpn) * SAR!
        ELSE
            R = WpnX%(HR.Wpn)
        END IF
        CIRCLE (HR.X, HR.Y), R, Colr%, , , SF * SAR!
    END IF
ELSEIF UXOs$(AV%) = "ON" THEN
    CIRCLE (HR.X, HR.Y), 25, Colr%, , , SAR!
END IF
HPtr% = HPtr% + 1
GET AHF%, HPtr%, HR
WEND
END SUB

*****
REM This SUB-PROGRAM determines the bomb stick starting and
REM ending point and draws a line between the two points
REM representing the stick.
*****

SUB PlotStick (X%, Y%, W, H, Phi, Colr%)
    X1 = FNXL(X%, W, H, Phi)
    Y1 = FNY1(Y%, W, H, Phi)
    X3 = FNXL(X%, W, H, Phi)
    Y3 = FNY3(Y%, W, H, Phi)

```

```

LINE (X1, Y1)-(X3, Y3), Colr%

END SUB

*****
REM This SUB-PROGRAM shows attack and hit file information
REM (File, attack, time of day, day of attack) on line 23.
*****

SUB PlotSubTitle (AV%, AW%, AAF%, AHF%)

  SHARED Attack$(), Hit$(), SRow%(), MaxViews%

  DIM STitle$(MaxViews%)

  A = LEN(Attack$(AV%))

  H = LEN(Hit$(AV%))

  IF A > 8 AND H > 5 THEN

    IF A > 38 THEN Attack$(AV%) = "ATTACKS:  F" +
      FNT$(AAF%) + "/MULTIPLE/*"

    IF H > 38 THEN Hit$(AV%) = "HITS:  F" + FNT$(AHF%) +
      "/MULTIPLE/*"

    STitle$(AV%) = Attack$(AV%) + " - " + Hit$(AV%)

  ELSEIF A > 8 THEN

    IF A > 78 THEN Attack$(AV%) = "ATTACKS:  F" +
      FNT$(AAF%) + "/MULTIPLE/*"

    STitle$(AV%) = Attack$(AV%)

  ELSEIF H > 5 THEN

    IF H > 78 THEN Hit$(AV%) = "HITS:  F" + FNT$(AHF%) +
      "/MULTIPLE/*"

    STitle$(AV%) = Hit$(AV%)

  ELSE

    STitle$(AV%) = " "

  END IF

  CALL PrintLine(SRow%(AW%), (39 - LEN(STitle$(AV%)) / 2),
    (STitle$(AV%)))

END SUB

```

```

*****
REM This SUB-PROGRAM takes the coordinates found in the
REM TARGETs text file and draws lines to represent
REM buildings, runways, and taxiways.
*****

```

```

SUB PlotTargets (NumTargets%)

```

```

    SHARED Tgt(), TgtColr%(), TgtFill%()

```

```

    FOR I% = 1 TO NumTargets%

```

```

        TgtType% = Tgt(I%, 9)

```

```

        Colr% = TgtColr%(TgtType%) ' Sets color based on
        ' target type.

```

```

        LINE (Tgt(I%, 1), Tgt(I%, 2))-(Tgt(I%, 3), Tgt(I%,
        4)), Colr%

```

```

        LINE (Tgt(I%, 3), Tgt(I%, 4))-(Tgt(I%, 5), Tgt(I%,
        6)), Colr%

```

```

        LINE (Tgt(I%, 5), Tgt(I%, 6))-(Tgt(I%, 7), Tgt(I%,
        8)), Colr%

```

```

        LINE (Tgt(I%, 7), Tgt(I%, 8))-(Tgt(I%, 1), Tgt(I%,
        2)), Colr%

```

```

        IF TgtFill%(TgtType%) = 1 THEN ' Determines if target
        ' gets filled.

```

```

            CALL GetBounds(I%, Colr%, XW%, YW%)

```

```

            IF (XW% > 1) AND (YW% > 1) THEN

```

```

                X = (Tgt(I%, 1) + Tgt(I%, 5)) / 2

```

```

                Y = (Tgt(I%, 2) + Tgt(I%, 6)) / 2

```

```

                PAINT (X, Y), Colr%, Colr%

```

```

            END IF

```

```

        END IF

```

```

    NEXT

```

```

END SUB

```

```

*****
REM This SUB-PROGRAM prints the title of the base being
REM simulated plus any active attack and hit files on the
REM top of the screen.
*****

SUB PlotTitle (BoldColr%, DefColr%, AAF%, AHF%)

    SHARED Title$, TgtFile$, WpnFile$, AttFile$(), HitFile$()

    Temp = 10 + LEN(Title$) + LEN(TgtFile$) + LEN(WpnFile$)

    FOR I% = 1 TO 2

        Temp = Temp + LEN(AttFile$(I%)) + LEN(HitFile$(I%))

    NEXT I%

    LOCATE 1, 40 - Temp / 2

    COLOR DefColr%

    PRINT Title$ + " - (" + TgtFile$ + "," + WpnFile$ + ",";

    IF AAF% = 1 THEN ' Checks to see if any active attack
        ' files.

        COLOR BoldColr%: PRINT AttFile$(1);

        COLOR DefColr%: PRINT "," + AttFile$(2) + ",";

    ELSE

        PRINT AttFile$(1) + ",";

        COLOR BoldColr%: PRINT AttFile$(2);

        COLOR DefColr%: PRINT ",";

    END IF

    IF AHF% = 1 THEN ' Checks to see if any active hit files.

        COLOR BoldColr%: PRINT HitFile$(1);

        COLOR DefColr%: PRINT "," + HitFile$(2) + ",";

    ELSE

        PRINT HitFile$(1) + ",";

        COLOR BoldColr%: PRINT HitFile$(2);

        COLOR DefColr%: PRINT ",";

    END IF

```

```

        END IF

    END SUB

    *****
    REM This SUB-PROGRAM is used to print error information on
    REM line 24. It is called from the Error traps in the main
    REM program.
    *****

    SUB PrintErrMsg (Num%, Msg$)

        BEEP

        CALL PrintLine(24, 1, ("ERROR # " + STR$(Num%) + ": " +
        Msg$))

        DO

            LOOP WHILE INKEY$ = ""

        END SUB

    *****
    REM This SUB-PROGRAM prints a line of information based on
    REM the memory variables input from other modules. For
    REM example the test string variable might contain a
    REM question asking for a user input.
    *****

    SUB PrintLine (Row%, Col%, text$)

        CALL ClrLine(Row%)

        LOCATE Row%, Col%

        PRINT text$;

    END SUB

    *****
    REM This SUB-PROGRAM prints the main menu on the screen at
    REM row 25.
    *****

    SUB PrintMenu (MenuColr%, DefColr%)

        READ NumOptns%, Row%, OptnsColr%

        READ Col%, Menu$

        COLOR MenuColr%

        CALL PrintLine(Row%, Col%, (Menu$))

```

```

COLOR OptnsColr%

FOR I% = 1 TO NumOptns%

    READ Col%, Menu$

    LOCATE Row%, Col%: PRINT Menu$;

NEXT I%

COLOR DefColr%

END SUB

*****
REM This SUB-PROGRAM reads attack text file which is in
REM  TSARINA card column format.
*****

SUB ReadNewAttacks (Path$, Name$, Ext$, NumAttacks%, AAF%)

    SHARED AttDay%(), AttHour%(), AttPtr%(), MaxAttacks%

    DIM AR AS AttRecordType

    Num% = AAF% + 3

    OPEN "I", #3, Path$ + Name$ + Ext$

    OPEN "R", Num%, Path$ + Name$ + ".$$$", LEN(AR)

    CALL PrintLine(24, 25, ("Reading Attack:      ATT_#:"))

    K% = 0    '# of DATA cards (attacks)

    J% = 0    '# of ATT cards

    WHILE NOT EOF(3)

        LINE INPUT #3, Card$

        IF LEFT$(Card$, 4) = "DATA" THEN

            K% = K% + 1

            IF K% > MaxAttacks% THEN ERROR 102

            AttDay%(K%, AAF%) = VAL(MID$(Card$, 29, 2))

            AttHour%(K%, AAF%) = VAL(MID$(Card$, 33, 4))

            AttPtr%(K%, AAF%) = J% + 1

            LOCATE 24, 41: PRINT K%;

```

```

ELSEIF LEFT$(Card$, 3) = "ATT" THEN
    AR.Num = K% ' Attack
    AR.Phi = VAL(MID$(Card$, 8, 3)) * 3.141592 / 180
' Heading
    AR.X = VAL(MID$(Card$, 13, 6)) ' X coord, DMPI
    AR.Y = VAL(MID$(Card$, 19, 6)) ' Y coord, DMPI
    AR.Bomb = VAL(MID$(Card$, 49, 6)) ' # of bombs
    AR.SLen = VAL(MID$(Card$, 55, 6)) ' Stick length
    AR.Wpn = VAL(MID$(Card$, 65, 2)) ' Weapon type
    IF AR.Bomb > 1 THEN
        AR.W = AR.SLen / 2
        AR.Inc = AR.SLen / (AR.Bomb - 1)
        AR.Ofst = AR.Inc / 2
    ELSE
        AR.W = 50
        AR.Inc = 0
        AR.Ofst = 0
    END IF
    FOR I% = 1 TO VAL(MID$(Card$, 5, 2)) ' Add an ATT
' card for each pass
        J% = J% + 1
        PUT Num%, , AR
        LOCATE 24, 53: PRINT J%;
    NEXT I%
END IF
WEND
CLOSE #3

```

```

    NumAttacks% = K%      ' Set number of attacks

END SUB

*****
REM This SUB-PROGRAM reads a hit text file which is output
REM  from TSARINA.
*****

SUB ReadNewHits (Path$, Name$, Ext$, NumHits%, NumTrials%,
AHF%)

    SHARED HitPtr%()

    DIM HR AS HitRecordType

    OPEN "I", #3, Path$ + Name$ + Ext$

    OPEN "R", AHF%, Path$ + Name$ + ".$$$", LEN(HR)

    CALL PrintLine(24, 25, ("Reading Attack:  Trial: Bomb:"))

    I% = 0      ' # of cases (attacks)

    J% = 0      ' # of trials

    K% = 0      ' # of impacts

    WHILE NOT EOF(3)

        LINE INPUT #3, Card$

        Temp$ = LEFT$(Card$, 6)

        Temp% = VAL(Temp$)

        IF Temp$ = " CASE:" THEN

            Atk% = VAL(MID$(Card$, 7, 4))

            Trl% = VAL(MID$(Card$, 19, 4))

            SELECT CASE Atk%

                CASE IS < I%

                    ERROR 103

                CASE I%

                    SELECT CASE Trl%

                        CASE IS <= J%

```

```

        ERROR 104

        CASE J% + 1 TO NumTrials%

            CALL FillHitPtr(I%, I%, J% + 1, Trl%,
                AHF%, K% + 1)

            J% = Trl%

        CASE ELSE

            ERROR 101

        END SELECT

        CASE I% + 1 TO NumHits%

            CALL FillHitPtr(I%, I%, J% + 1, NumTrials%,
                AHF%, K% + 1)

            CALL FillHitPtr(I% + 1, Atk% - 1, 1,
                NumTrials%, AHF%, K% + 1)

            I% = Atk%

            CALL FillHitPtr(I%, I%, 1, Trl%, AHF%, K% +
                1)

            J% = Trl%

        CASE ELSE

            ERROR 100

        END SELECT

        LOCATE 24, 41: PRINT I%;

        LOCATE 24, 53: PRINT J%;

    ELSEIF Temp% <> -32000 THEN

        K% = K% + 1

        HR.Atk = I%

        HR.Tr1 = J%

        HR.X = VAL(MID$(Card$, 1, 6))

        HR.Y = VAL(MID$(Card$, 7, 6))

        HR.Wpn = VAL(MID$(Card$, 13, 6))

```

```

        HR.UXO = VAL(MID$(Card$, 19, 6))
        HR.Phi = VAL(MID$(Card$, 25, 6))
        HR.Alt = VAL(MID$(Card$, 31, 6))

        PUT AHF%, , HR

    END IF

    LOCATE 24, 64: PRINT K%;

WEND

CLOSE #3

END SUB

*****
REM This SUB-PROGRAM reads a target text file which is in
REM  TSARINA card column format.
*****

SUB ReadNewTargets (Path$, Name$, Ext$, NumTargets%, XMax)

    SHARED Tgt(.), MaxTargets%

    ON ERROR GOTO TgtFileNameError

    XMax = 0

    I% = 0      ' # of TGT cards

    LOCATE 14, 25: PRINT "Reading Target Number:"; ' Prints
    ' message on screen.

    OPEN "I", #1, Path$ + Name$ + Ext$

    WHILE NOT EOF(1) AND I% < MaxTargets%

        LINE INPUT #1, Card$

        IF LEFT$(Card$, 3) = "TGT" THEN

            I% = I% + 1

            LOCATE 14, 47: PRINT I%;

            H = VAL(MID$(Card$, 19, 6)) ' Reads height of
            ' target.

            W = VAL(MID$(Card$, 25, 6)) ' Reads width of
            ' target.

```

```

    Phi = VAL(MID$(Card$, 34, 3)) * 3.141592 / 180
    ' Reads heading (relative to 0 degs.) of target and
    ' converts it to radians.

    Tgt(I%, 1) = VAL(MID$(Card$, 7, 6)) ' X-coordinate
    ' of target.

    Tgt(I%, 2) = VAL(MID$(Card$, 13, 6)) ' Y-coordinate
    ' of target.

    Tgt(I%, 3) = Tgt(I%, 1) + H * SIN(Phi) ' The rest
    ' computes the remainig three coordinates based on
    ' above inputs.

    Tgt(I%, 4) = Tgt(I%, 2) + H * COS(Phi)

    Tgt(I%, 5) = Tgt(I%, 3) + W * COS(Phi)

    Tgt(I%, 6) = Tgt(I%, 4) - W * SIN(Phi)

    Tgt(I%, 7) = Tgt(I%, 5) - H * SIN(Phi)

    Tgt(I%, 8) = Tgt(I%, 6) - H * COS(Phi)

    Tgt(I%, 9) = VAL(MID$(Card$, 41, 2)) ' Reads target
    ' type.

    IF Tgt(I%, 1) + W > XMax THEN XMax = Tgt(I%, 1) + W
    ' Sets XMax each time it reads a target and determines
    ' final maximum X value.

END IF

WEND

CLOSE #1

NumTargets% = I%

END SUB

*****
REM This SUB-PROGRAM reads files with .Sl$ and .SS$
REM extensions. These files are in binary format which
REM were created after reading the initial Attack files in
REM TSARINA format.
*****

SUB ReadOldAttacks (Path$, Name$, NumAttacks%, AAF%)

    SHARED AttDay%(), AttHour%(), AttPtr%()

    DIM AR AS AttRecordType

```

```

Num% = AAF% + 3

OPEN "R", Num%, Path$ + Name$ + ".$$$", LEN(AR)

CALL PrintLine(24, 25, ("Reading Attack:      ATT_#:"))

OPEN "I", #3, Path$ + Name$ + ".$l$"

INPUT #3, NumAttacks%

FOR I% = 1 TO NumAttacks%

    INPUT #3, AttPtr%(I%, AAF%), AttDay%(I%, AAF%),
    AttHour%(I%, AAF%)

    LOCATE 24, 41: PRINT I%;

    LOCATE 24, 53: PRINT AttPtr%(I%, AAF%);

NEXT I%

CLOSE #3

END SUB

*****
REM This SUB-PROGRAM reads files with .l$ and .$$$
REM extensions. These files are in binary format which
REM were created after reading the initial Hit files in
REM TSARINA format.
*****

SUB ReadOldHits (Path$, Name$, NumHits%, NumTrials%, AHF%)

    SHARED HitPtr%()

    DIM HR AS HitRecordType

    OPEN "R", AHF%, Path$ + Name$ + ".$$$", LEN(HR)

    CALL PrintLine(24, 25, ("Reading Attack:      Trial:
Bomb:"))

    OPEN "I", #3, Path$ + Name$ + ".$l$"

    INPUT #3, NumHits%, NumTrials%

    FOR I% = 1 TO NumHits%

        FOR J% = 1 TO NumTrials%

            INPUT #3, HitPtr%(I%, J%, AHF%)

            LOCATE 24, 41: PRINT I%;

```

```

        LOCATE 24, 53: PRINT J%;

        LOCATE 24, 64: PRINT HitPtr%(I%, J%, AHF%);

    NEXT J%

NEXT I%

CLOSE #3

END SUB

*****
REM This SUB-PROGRAM reads files with $.1$ and $.$$
REM extensions. These files are in binary format which
REM were created after reading the initial Target files in
REM TSARINA format.
*****

SUB ReadOldTargets (Path$, Name$, NumTargets%, XMax)

    SHARED Tgt()

    ON ERROR GOTO TgtFileNameError

    OPEN "I", #1, Path$ + Name$ + ".$1$"

        INPUT #1, NumTargets%, XMax

    CLOSE #1

    DEF SEG = VARSEG(Tgt(1, 1))

    BLOAD Path$ + Name$ + ".$$$", VARPTR(Tgt(1, 1))

END SUB

*****
REM This SUB-PROGRAM is used to redraw the active window
REM whenever there are changes made to the inputs of that
REM window.
*****

SUB ReDrawWindow (NumTargets%, AV%, AW%, NAF%, NHF%)

    SHARED A(), B(), C(), D()

    CLS

    WINDOW (A(AV%), B(AV%))-(C(AV%), D(AV%))

    CALL PlotTargets(NumTargets%)

    CALL SaveWindow(AW%, AV%)

```

```

CALL PlotAttacks(NAF%, AV%)

CALL PlotHits(NHF%, AV%)

CALL PlotGrid(AV%, AW%)

END SUB

*****
REM This SUB-PROGRAM resets various controls in the main
REM program.
*****

SUB ResetControl (BoldColr%, DefColr%, AAF%, NAF%, AHF%,
NHF%, AV%, AW%)

    RESTORE ResetMenu

    CALL PrintMenu(BoldColr%, DefColr%)

    DO

        Optn$ = GetOptn$(23, 35, "RESET? ")

        SELECT CASE Optn$

            CASE "M", "m" ' Returns coordinates that match.

                CALL ResetMatching(NAF%, NHF%, AV%, AW%)

                EXIT DO

            CASE "S", "s" ' Returns to the initial starting
            ' coordinates.

                CALL ResetStartup(NAF%, NHF%, AV%, AW%)

                EXIT DO

            CASE "V", "v" ' Resets the graphics area to the
            ' maximum size.

                CALL ResetView(BoldColr%, DefColr%, AAF%, NAF%,
                AHF%, NHF%, AV%, AW%)

                RESTORE MainMenu

                CALL PrintMenu(BoldColr%, DefColr%)

                EXIT DO

            CASE "X", "x"

                EXIT DO

```

```

        CASE ELSE

            BEEP

        END SELECT

    LOOP

        CALL ClrLine(24)

    END SUB

    *****
    REM This SUB-PROGRAM determines active windows and sets
    REM original graphics coordinates within each window.
    *****

    SUB ResetMatching (NAF%, NHF%, AV%, AW%)

        SHARED NumTargets%, A(), B(), C(), D(), VY%(), S1%(),
        S2%()

        Temp% = 3 - AV%

        A(AV%) = A(Temp%)

        B(AV%) = B(Temp%)

        C(AV%) = C(Temp%)

        D(AV%) = D(Temp%)

        SELECT CASE AW%

            CASE 1

                CLS

                WINDOW (A(AV%), B(AV%))-(C(AV%), D(AV%)) ' Defines
                ' graphics area.

                CALL PlotTargets(NumTargets%)

                CALL SaveWindow(AW%, AV%)

            CASE 2

                WINDOW SCREEN (20, VY%(2, 1))-(620, VY%(2, 2))
                ' Defines graphics area.

                PUT (20, VY%(2, 1)), S2%, PSET ' Draws on the
                ' screen a graphics image stored in specified array.

                GET (20, VY%(2, 1))-(620, VY%(2, 2)), S1% ' Stores

```

```

' a graphics image into an array.
  WINDOW (A(AV%), B(AV%))-(C(AV%), D(AV%))

CASE 3

  WINDOW SCREEN (20, VY%(3, 1))-(620, VY%(3, 2))

  PUT (20, VY%(3, 1)), S1%, PSET

  GET (20, VY%(3, 1))-(620, VY%(3, 2)), S2%

  WINDOW (A(AV%), B(AV%))-(C(AV%), D(AV%))

END SELECT

CALL PlotAttacks(NAF%, AV%)

CALL PlotHits(NHF%, AV%)

CALL PlotGrid(AV%, AW%)

END SUB

*****
REM This SUB-PROGRAM resets the split coordinates to be used
REM when using split screens.
*****

SUB ResetSplitCoord (AV%) .

  SHARED VY%(), A(), B(), C(), D()

  D(AV%) = B(AV%) + 4 * (D(AV%) - B(AV%)) / 3

  Temp = (D(AV%) - B(AV%)) / FND(20, 620, VY%(1, 1), VY%(1, 2))

  Temp = (C(AV%) - A(AV%)) - Temp

  A(AV%) = A(AV%) + Temp / 2

  C(AV%) = C(AV%) - Temp / 2

END SUB

*****
REM This SUB-PROGRAM returns the screens to the original
REM coordinates used prior to zooming or panning.
*****

SUB ResetStartup (NAF%, NHF%, AV%, AW%)

  SHARED A(), B(), C(), D(), XMax, VY%(), NumTargets%

```

```

A(AV%) = 0
B(AV%) = 0
C(AV%) = XMax
D(AV%) = C(AV%) * FND(20, 620, VY%(1, 1), VY%(1, 2))
IF AW% > 1 THEN
    CALL SetSplitCoord(AV%, AW%)
END IF

CALL ReDrawWindow(NumTargets%, AV%, AW%, NAF%, NHF%)

END SUB

*****
REM This SUB-PROGRAM resets the graphics area to its maximum
REM size.
*****

SUB ResetView (BoldColr%, DefColr%, AAF%, NAF%, AHF%, NHF%,
AV%, AW%)

    SHARED VMax, SColr%()

    VIEW (0, 0)-(639, VMax) ' VMax depends on the type of
        ' screen computer has.

    CLS

    CALL PlotTitle(BoldColr%, DefColr%, AAF%, AHF%)

    IF AW% > 1 THEN
        AW% = 5 - AW%
        AV% = 3 - AV%

        CALL PlotBorder(AW%, AV%, DefColr%)
        CALL PlotSubTitle(AV%, AW%, AAF%, AHF%)
        CALL RestoreWindow(AW%, AV%)
        CALL PlotAttacks(NAF%, AV%)
        CALL PlotHits(NHF%, AV%)
        CALL PlotGrid(AV%, AW%)
        AW% = 5 - AW%
    
```

```

        AV% = 3 - AV%

    END IF

    CALL PlotBorder(AW%, AV%, SColr%(AV%))

    CALL PlotSubTitle(AV%, AW%, AAF%, AHF%)

    CALL RestoreWindow(AW%, AV%)

    CALL PlotAttacks(NAF%, AV%)

    CALL PlotHits(NHF%, AV%)

    CALL PlotGrid(AV%, AW%)

END SUB

*****
REM This SUB-PROGRAM restores the current active windows to
REM graphics arrays.
*****

SUB RestoreWindow (AW%, AV%)

    SHARED S1%(), S2%(), VY%(), A(), B(), C(), D()

    WINDOW SCREEN (20, VY%(AW%, 1))-(620, VY%(AW%, 2))

    SELECT CASE AW%

        CASE 1

            PUT (20, VY%(AW%, 1)), S1%, PSET

            PUT (20, 146), S2%, PSET

        CASE 2

            PUT (20, VY%(AW%, 1)), S1%, PSET

        CASE 3

            PUT (20, VY%(AW%, 1)), S2%, PSET

    END SELECT

    WINDOW (A(AV%), B(AV%))-(C(AV%), D(AV%))

END SUB

```

```

*****
REM This SUB-PROGRAM saves current window to graphic arrays
REM so they can be recalled later.
*****

```

```

SUB SaveWindow (AW%, AV%)

```

```

    SHARED S1%(), S2%(), VY%(), A(), B(), C(), D()

```

```

    WINDOW SCREEN (20, VY%(AW%, 1))-(620, VY%(AW%, 2))

```

```

    SELECT CASE AW%

```

```

        CASE 1

```

```

            GET (20, VY%(1, 1))-(620, 145), S1%

```

```

            GET (20, 146)-(620, VY%(1, 2)), S2%

```

```

        CASE 2

```

```

            GET (20, VY%(2, 1))-(620, VY%(2, 2)), S1%

```

```

        CASE 3

```

```

            GET (20, VY%(3, 1))-(620, VY%(3, 2)), S2%

```

```

    END SELECT

```

```

    WINDOW (A(AV%), B(AV%))-(C(AV%), D(AV%))

```

```

END SUB

```

```

*****
REM This SUB-PROGRAM determines the initial split
REM coordinates to be used whenever the user decides to
REM view two windows on the screen.
*****

```

```

SUB SetSplitCoord (AV%, AW%)

```

```

    SHARED VY%(), A(), B(), C(), D()

```

```

    D(AV%) = B(AV%) + .75 * (D(AV%) - B(AV%))

```

```

    Temp = (D(AV%) - B(AV%)) / FND(20, 620, VY%(AW%, 1),
    VY%(AW%, 2))

```

```

    Temp = Temp - (C(AV%) - A(AV%))

```

```

    A(AV%) = A(AV%) - Temp / 2

```

```

        C(AV%) = C(AV%) + Temp / 2

END SUB

*****
REM This SUB-PROGRAM determines the weapon status for each
REM weapon type.
*****

SUB SetWpnStat (Stat%)

'   Stat% = 0 for Wpn display off, = 1 for Wpn display on

    SHARED WpnStat%(), MaxWpnTypes%

    FOR I% = 1 TO MaxWpnTypes%

        WpnStat%(I%) = Stat%

    NEXT I%

END SUB

*****
REM This SUB-PROGRAM is used to split the graphics area in
REM half to allow the user to view two windows at once.
*****

SUB SplitControl (DefColr%, AAF%, NAF%, AHF%, NHF%, AV%,
AW%)

    SHARED VY%(), SColr%()

    VIEW (19, VY%(1, 1) - 1)-(621, VY%(1, 2) + 1)

    CLS

    IF AW% = 1 THEN

        AW% = 4 - AV%

        AV% = 3 - AV%

        CALL SetSplitCoord(AV%, AW%)

        CALL DrawWindow(AAF%, NAF%, AHF%, NHF%, AW%, AV%,
        DefColr%)

        AW% = 5 - AW%

        AV% = 3 - AV%

        CALL SetSplitCoord(AV%, AW%)

```

```

        CALL DrawWindow(AAF%, NAF%, AHF%, NHF%, AW%, AV%,
        SColr%(AV%))

ELSE

    AW% = 1

    AV% = 3 - AV%

    CALL ResetSplitCoord(AV%)

    AV% = 3 - AV%

    CALL ResetSplitCoord(AV%)

    CALL DrawWindow(AAF%, NAF%, AHF%, NHF%, AW%, AV%,
    SColr%(AV%))

END IF

END SUB

*****
REM This SUB-PROGRAM switches the file that is currently
REM active.  There can be up to two files (Attack and Hit)
REM open at the same time but the user can only view one
REM file at a time.  The active files are displayed in bold
REM white on the title line.
*****

SUB ToggleActFile (AF%, NF%)

    IF NF% = 2 THEN

        AF% = 3 - AF%

    ELSE

        BEEP

    END IF

END SUB

*****
REM This SUB-PROGRAM changes the color of the background.
REM Turning background colors off allows the user to see
REM the attacks and hits more clearly.
*****

SUB ToggleBGrd (BGrd$)

    IF BGrd$ = "ON" THEN

```

```

        CALL ChangePalette(0, 3)

        BGrd$ = "OFF"

    ELSE

        CALL ChangePalette(0, 1)

        BGrd$ = "ON"

    END IF

END SUB

*****
REM This SUB-PROGRAM determines what the users wants to turn
REM on or off by toggling certian program characteristics.
*****

SUB ToggleControl (BoldColr%, DefColr%, NAF%, AAF%, NHF%,
AHF%, AV%, AW%, BGrd$, FGrd$)

    RESTORE ToggleMenu

    CALL PrintMenu(BoldColr%, DefColr%)

    DO

        Optn$ = GetOptn$(23, 35, "TOGGLE? ")

        SELECT CASE Optn$

            CASE "A", "a"      ' Changes active attack file.

                CALL ToggleActFile(AAF%, NAF%)

                CALL PlotTitle(BoldColr%, DefColr%, AAF%, AHF%)

                EXIT DO

            CASE "H", "h"      ' Changes active hit file.

                CALL ToggleActFile(AHF%, NHF%)

                CALL PlotTitle(BoldColr%, DefColr%, AAF%, AHF%)

                EXIT DO

            CASE "B", "b"      ' Changes background color.

                CALL ToggleBGrd(BGrd$)

                EXIT DO

```

```

CASE "F", "f"    ' Changes foreground color.

    CALL ToggleFGrd(FGrd$)

    EXIT DO

CASE "G", "g"    ' Turns the grid on or off.

    CALL ToggleGrid(NAF%, NHF%, AW%, AV%)

    EXIT DO

CASE "U", "u"    ' Shows the UXOs on the screen.

    CALL ToggleUXOs(NAF%, NHF%, AW%, AV%)

    EXIT DO

CASE "E", "e"    ' Turns on or off the effects
' (highlights certain hits or targets on the screen).

    CALL ToggleEffects(NAF%, NHF%, AW%, AV%)

    EXIT DO

CASE "S", "s"    ' Switches the active views.

    CALL ToggleScreen(AAF%, NAF%, AHF%, NHF%, AW%,
    AV%, DefColr%)

    EXIT DO

CASE "X", "x"

    EXIT DO

CASE ELSE

    BEEP

END SELECT

LOOP

CALL ClrLine(24)

END SUB

*****
REM This SUB-PROGRAM turns on the effects for displaying
REM attacks, hits, or the grid.
*****

SUB ToggleEffects (NAF%, NHF%, AW%, AV%)

```

```

    SHARED ECov$()

    IF ECov$(AV%) = "ON" THEN
        ECov$(AV%) = "OFF"

        CALL RestoreWindow(AW%, AV%)
    ELSE
        ECov$(AV%) = "ON"
    END IF

    CALL PlotAttacks(NAF%, AV%)

    CALL PlotHits(NHF%, AV%)

    CALL PlotGrid(AV%, AW%)

END SUB

*****
REM This SUB-PROGRAM changes the foreground colors based on
REM weapon status. Turning foreground colors off and then
REM using the function keys allows the users to clearly see
REM individual weapon types.
*****

SUB ToggleFGrd (FGrd$)

    SHARED WpnStat%()

    Temp% = 0

    FOR I% = 1 TO 10
        Temp% = Temp% OR WpnStat%(I%)
    NEXT I%

    IF Temp% = 1 THEN
        CALL ChangePalette(8, 3)

        CALL SetWpnStat(0)

        FGrd$ = "OFF"
    ELSE
        CALL ChangePalette(8, 2)

        CALL SetWpnStat(1)
    END IF

```

```

        FGrd$ = "ON"

    END IF

END SUB

*****
REM This SUB-PROGRAM turns on and off the grid system.
*****

SUB ToggleGrid (NAF%, NHF%, AW%, AV%)
    SHARED Grid$()

    IF Grid$(AV%) = "ON" THEN

        Grid$(AV%) = "OFF"

        CALL RestoreWindow(AW%, AV%)

        CALL PlotAttacks(NAF%, AV%)

        CALL PlotHits(NHF%, AV%)

    ELSE

        Grid$(AV%) = "ON"

        CALL PlotGrid(AV%, AW%)

    END IF

END SUB

*****
REM This SUB-PROGRAM changes which screen is active by
REM changing the color of the border around the screen.
*****

SUB ToggleScreen (AAF%, NAF%, AHF%, NHF%, AW%, AV%,
    DefColr%)

    SHARED SColr%()

    IF AW% = 1 THEN

        CLS

        AV% = 3 - AV%

        CALL DrawWindow(AAF%, NAF%, AHF%, NHF%, AW%, AV%,
            SColr%(AV%))

    ELSE

```

```

        CALL PlotBorder(AW%, AV%, DefColr%)

        AW% = 5 - AW%

        AV% = 3 - AV%

        CALL PlotBorder(AW%, AV%, SColr%(AV%))

    END IF

END SUB

*****
REM This SUB-PROGRAM determines whether the unexploded
REM ordnance is shown on screen.
*****

SUB ToggleUXOs (NAF%, NHF%, AW%, AV%)

    SHARED UXOs$( )

    IF UXOs$(AV%) = "ON" THEN

        UXOs$(AV%) = "OFF"

        CALL RestoreWindow(AW%, AV%)

        CALL PlotAttacks(NAF%, AV%)

        CALL PlotHits(NHF%, AV%)

        CALL PlotGrid(AV%, AW%)

    ELSE

        UXOs$(AV%) = "ON"

        CALL PlotHits(NHF%, AV%)

    END IF

END SUB

*****
REM This SUB-PROGRAM changes the colors of the weapons
REM displayed on the screen.
*****

SUB ToggleWpn (WpnNum%)

    SHARED WpnStat%( ), WpnColr%( )

    IF WpnStat%(WpnNum%) = 1 THEN

```

```

        WpnStat%(WpnNum%) = 0

        PALETTE WpnColr%(WpnNum%), 4

ELSE

        WpnStat%(WpnNum%) = 1

        PALETTE WpnColr%(WpnNum%), WpnColr%(WpnNum%) + 48

END IF

END SUB

*****
REM This SUB-PROGRAM writes a binary file of the TSARINA
REM format text file to allow for a quicker display of
REM inputs the next time program is called.
*****

SUB WriteAttacks (Path$, Name$, NumAttacks%, AAF%)

    SHARED AttDay%(), AttHour%(), AttPtr%()

    OPEN "O", #3, Path$ + Name$ + ".$1$"

    WRITE #3, NumAttacks%

    FOR I% = 1 TO NumAttacks%

        WRITE #3, AttPtr%(I%, AAF%), AttDay%(I%, AAF%),
            AttHour%(I%, AAF%)

    NEXT I%

    CLOSE #3

END SUB

*****
REM This SUB-PROGRAM writes a binary file of the TSARINA
REM format text file to allow for a quicker display of
REM inputs the next time program is called.
*****

SUB WriteHits (Path$, Name$, NumHits%, NumTrials%, AHF%)

    SHARED HitPtr%()

    OPEN "O", #3, Path$ + Name$ + ".$1$"

    WRITE #3, NumHits%, NumTrials%

    FOR I% = 1 TO NumHits%

```

```

    FOR J% = 1 TO NumTrials%
        WRITE #3, HitPtr%(I%, J%, AHF%)
    NEXT J%
NEXT I%
CLOSE #3

END SUB

*****
REM This SUB-PROGRAM writes a binary file of the TSARINA
REM format text file to allow for a quicker display of
REM inputs the next time program is called.
*****

SUB WriteTargets (Path$, Name$, NumTargets%, XMax)

    SHARED Tgt()

    OPEN "O", #1, Path$ + Name$ + ".$1$"

        WRITE #1, NumTargets%, XMax

    CLOSE #1

    DEF SEG = VARSEG(Tgt(1, 1))

    BSAVE Path$ + Name$ + ".$$$", VARPTR(Tgt(1, 1)), 36000

END SUB

*****
REM This SUB-PROGRAM changes the value of the coordinate
REM system to allow the user to get a closer view of
REM various sections of the base.
*****

SUB ZoomControl (BoldColr%, DefColr%, ZF%, AV%, AW%, NAF%,
NHF%)

    SHARED A(), C(), NumTargets%

    STATIC Temp$

    RESTORE ZoomMenu

    CALL PrintMenu(BoldColr%, DefColr%)

    DO

        Optn$ = GetOptn$(23, 36, "ZOOM? ")

```

```

SELECT CASE Optn$
    CASE "I", "i"
        Temp = C(AV%) - A(AV%)
        IF Temp <= 2 * ZF% THEN ZF% = Temp / 5
        CALL ZoomCoordinates(AV%, AW%, (ZF%), (ZF%),
            (-ZF%))
        CALL ReDrawWindow(NumTargets%, AV%, AW%, NAF%,
            NHF%)
        EXIT DO
    CASE "O", "o"
        CALL ZoomCoordinates(AV%, AW%, (-ZF%), (-ZF%),
            (ZF%))
        CALL ReDrawWindow(NumTargets%, AV%, AW%, NAF%,
            NHF%)
        EXIT DO
    CASE "C", "c"
        Temp$ = "Old Zoom Factor =" + STR$(ZF%) + " New
        Zoom Factor ="
        ZF% = GetIData%(23, (Temp$), 0, 10000)
    CASE "X", "x"
        EXIT DO
    CASE ELSE
        BEEP
END SELECT

LOOP

CALL ClrLine(24)

END SUB

```

```
*****
REM This SUB-PROGRAM determines the new coordinate values
REM based on whether the user wants zoom in or out.
*****
```

```
SUB ZoomCoordinates (AV%, AW%, AF%, BF%, CF%)
```

```
    SHARED A(), B(), C(), D(), VY%()
```

```
    A(AV%) = A(AV%) + AF%
```

```
    B(AV%) = B(AV%) + BF% * FND(20, 620, VY%(AW%, 1),  
    VY%(AW%, 2))
```

```
    C(AV%) = C(AV%) + CF%
```

```
    D(AV%) = B(AV%) + (C(AV%) - A(AV%)) * FND(20, 620,  
    VY%(AW%, 1), VY%(AW%, 2))
```

```
END SUB
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 1990	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE AN AIR BASE VULNERABILITY ASSESSMENT ANALYSIS TOOL FOR U.S. AIR FORCE WAR PLANNERS VOLUME II: TECHNICAL REFERENCE MANUAL			5. FUNDING NUMBERS	
6. AUTHOR(S) Richard M. Cockley, Captain, USAF				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583			8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GLM/LSM/90S-12	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) BasePlot's, a pre-and post-processor for TSARINA, Volume II: Technical Reference Manual contains three chapters. Chapter I, Data Dictionary, contains a description of data in BasePlot. Chapter II, Definition Sub-Programs and Sub-Functions, contains a brief description of each individual sub-program or sub-function. Chapter III, Program Documentation, contains QuickBASIC 4.5 program code written for BasePlot. Application and BasePlot's User's Manual are documented in Volume I: Development and User's Manual.				
14. SUBJECT TERMS Air Base Operability, Graphics Pre-Processor, Vulnerability, Graphics Post-Processor, Air Force Facilities, Bomb Damage, Conventional Warfare, Simulation Models			15. NUMBER OF PAGES 107	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	