MAP PLOTTING SYSTEM FOR THE SUMMASKETCH GRAPHICS TABLET

GRADUATE PROJECT REPORT

BY

DARRIN GATHRIGHT

COMMITTEE MEMBERS

Dr. Roy Ellzey,
Chairman

Dr. Alan Berkebile,
Member

Dr. Patrick Michaud,
Member
ABSTRACT

This system provides a software environment for the use of a Summasketch graphics tablet on an IBM-compatible PC. It allows a geologist to measure and record the positions of up to 500 two-dimensional or three-dimensional points on any number of maps. The geologist measures the two-dimensional positions of the points using the tablet cursor, and if plotting in three dimensions, receives prompts for the altitude of each point once two-dimensional plotting is complete. The user is able to specify a scale for each map, and have all measurements adjusted accordingly. Once the data for a map has been collected, it is saved in a user specified file formatted for use by other graphics applications. System functions such as communication with the tablet and data conversion are virtually invisible to the user.
# TABLE OF CONTENTS

Introduction ........................................ 4
Environment........................................ 6
User Procedure........................................ 7
File Descriptions..................................... 8
Module Descriptions................................... 9
Summary............................................... 21
Program Listing...................................... Appendix A
Module Flowchart..................................... Appendix B
Test Results......................................... Appendix C
INTRODUCTION

Currently, Dr. Berkebile and other members of the geology department produce topographic maps (three-dimensional) or profiles (two-dimensional) from plots of points as follows:

1) The two-dimensional (X and Y) coordinates of the points are measured by hand. These coordinates are based on an origin point in the lower left corner of the map, thus all possible values of X and Y are positive. As the X and Y coordinates for a point are only useful relative to the X and Y coordinates for other points, the precise location of the origin is not important. However, it is important that the position of the origin is constant relative to all points on a given map.

2) These two-dimensional coordinates are multiplied by the number of feet per inch (i.e. the scale of the map) to convert the values to units of feet.

3) If plotting in three-dimensions, the altitude (Z coordinate) of each point is recorded along with the two-dimensional (X and Y) coordinates.

4) The point coordinates are entered from the keyboard into an ASCII text file in an appropriate format for use by other applications.

5) A graphics software product, usually a package called Surfer, is used to draw topographic maps or profiles from the collected point coordinates (see Surfer Version 4 Reference Guide, 1987).

This system automates steps one through four, above, of the process of producing topographic maps or profiles from point maps. Step one (measuring the X and Y coordinates of the points) is handled using a Summasketch graphics tablet (see MM 1201 and MM 961 Data Tablets Technical Reference, 1984). The geologist places the map on the tablet, then measures the two-dimensional coordinates of the points using the tablet cursor. This reduces the possibility of invalid data being entered by eliminating manual point measurement. Step two (converting the coordinates to scale) is
performed by the system. Before beginning to plot a map, the geologist is prompted to enter the number of feet represented by one inch. All coordinates plotted using the tablet are adjusted to the specified scale, thus eliminating errors associated with manual calculations. If plotting in three-dimensions, the geologist is prompted to enter the altitude for each point, thus automating step three (recording the Z coordinate of each point). Once all plotting for a map is complete, step four (writing the information to a file) is performed by the system. The user need only specify the name of the file to use. The data is written as ASCII text in a format suitable for use by other applications.
ENVIRONMENT

This system operates on an IBM-compatible personal computer running PC or MS DOS. It is designed to communicate, through the serial port of the computer, with a Summasketch II graphics tablet using a four-button cursor. Default factory settings for baud, parity and stop bits are assumed for the tablet.
USER PROCEDURE

Because the system exists as a stand-alone, executable module, the user simply enters the name of the program to begin. Once the program is started, the user can plot any number of maps of up to 500 two-dimensional or three-dimensional points per map.

Although it is not required, it is recommended that the user sequentially number all points on a map before beginning to plot. Since all system references to a point are by its sequential point number, this makes keeping track of what has and hasn't been plotted much easier.

The origin of the tablet is defined as the lower left corner of the mapping area. As when plotting by hand, the position of the origin must be constant relative to all points on a map (i.e. the position of the map over the tablet must be fixed). Thus it is recommended that the user tape the map firmly to the tablet surface.

For each map plotted, the user is prompted to enter the following parameters before beginning to plot:
1) The number of feet represented by one inch according to the scale of the map.
2) The number of dimensions to be plotted (either two or three).
3) The name of the file to use for saving the collected data. If the specified file already exists, the user must choose to overwrite the file, append the data to the file, or choose a different file.

For each point on the map, the user will be prompted to position the cursor over the current point (according to the sequential order of the points) and press cursor button 1. The user can then press button 1 again to proceed to the next point, button 2 to remeasure the current point, or button 3 to quit plotting. If
plotting in three dimensions, the user will be prompted to enter
the altitude for each point once two-dimensional plotting is
complete. After a map has been plotted, the user will be prompted
to choose to plot another map or return to DOS.
FILE DESCRIPTIONS

The system uses an ASCII text file to store the collected point information so that it may be used by other graphics applications. The file contains up to 500 rows of data per map, such that there is one row per point. If plotting a three-dimensional point map, each row consists of three columns that represent the X, Y, and Z coordinates of the points, respectively. If plotting a two-dimensional point map, each row consists of two columns that represent the X and Y coordinates of the points, respectively. In either case, each column consists of an eleven character number, precise to the hundredths position, right justified, padded with blanks, and separated from the other columns by three spaces.
MODULE DESCRIPTIONS

MainMod: MainMod is the controlling module for the system. It performs the necessary module calls and the passing of variables for the system to execute. MainMod performs the following steps:

1) Call InitCom to establish a session with the graphics tablet.
2) Call StartPrompt to display the initial program prompt.
3) Call GetScale to obtain the scale for the current map.
4) Call GetDimen to obtain the dimensions for the current map.
5) Call GetFileName to obtain the name of the file to use for saving the collected information and to open the file in the appropriate mode.
6) Call XYStartPrompt to display a prompt to begin plotting in two dimensions.
7) Assign a value of one to the variable representing the number of the current point.
8) Assign a value of false to the boolean variable which controls the loop for plotting X and Y (step 9).
9) Perform the following steps until the user chooses to stop (i.e. until the boolean variable which controls the loop for plotting X and Y is equal to true):
   9.1) Call MeasureXY to obtain the X and Y coordinates of the current point.
   9.2) Call StoreXY to save the X and Y coordinates for the current point and update the number of the current point.
10) Assign the value of the variable representing the number of the current point to the variable representing the total number of points.
11) If plotting in three dimensions (i.e. if the variable representing the number of dimensions is equal to three), perform the following steps:
   11.1) Call ZStartPrompt to display a prompt to begin entering Z coordinates.
   11.2) Assign a value of one to the variable representing the number of the current point.
11.3) For each point, perform the following steps (i.e. while the number of the current point is not greater than the total number of points):

11.3.1) Call GetZ to obtain the Z coordinate for the current point.

11.3.2) Call StoreZ to save the Z coordinate for the current point and update the number of the current point.

11.4) Call ZEndPrompt to display a prompt that the Z coordinate has been entered for all points.

12) Call SaveToFile to write the collected information to a file.
13) Call EndPrompt to allow the user to choose to plot another map (return to step 2) or return to DOS.

InitCom: This module sets the computer's serial port to the proper configuration and initializes communication with the tablet.
INPUT: There is no input for this module.
PROCESSING: First, InitCom configures the serial port to the proper settings for communication with the tablet. Next, using the SendByte module, InitCom resets the tablet so a new session may be initiated. Once the tablet is reset, InitCom uses the SendByte module to initialize the tablet to the desired report rate and report mode. InitCom then uses the SendByte module to issue a trigger command to the tablet, after which ChkTab is called to determine if the tablet is responding. If the tablet is not responding, an appropriate message is displayed and a flag is set to return to DOS.

OUTPUT: Output for the InitCom module consists of single-byte, tablet commands that are passed to the SendByte module. The following commands, in hexadecimal form, are used by InitCom:
1) Hexadecimal 00, resets all tablet parameters to the default factory settings.
2) Hexadecimal 42, sets the tablet to point mode. In this mode, the tablet issues one five byte report when a cursor button is pressed.
3) Hexadecimal 54, sets the report rate of the tablet.
4) Hexadecimal 50, causes the tablet to transmit a response.
InitCom also returns a boolean variable to MainMod, the value
of which determines if the program continues or if control returns to
DOS.

StartPrompt: This module display the initial prompt for the system.
INPUT: There is no input for this module.
PROCESSING: StartPrompt first sets control-break on so that the
user can terminate the program at any point. Next, the initial
program prompt is displayed.
OUTPUT: The following text is displayed on the screen:

<<< Press control-break at any point to return to DOS >>>

Please enter the following parameters for the current map:

GetScale: The GetScale module allows the user to specify the scale
for the current map.
INPUT: The user is prompted to enter the number of feet per inch.
This number can be any valid floating point value.
PROCESSING: If the input operation is unsuccessful, the user is
prompted until a valid response is entered. The number of feet per
inch, along with the tablet resolution of 500 lines per inch, is
then used to calculate the number of feet per line of tablet
resolution.
OUTPUT: GetScale returns a floating point variable representing
the number of feet per line of tablet resolution to MainMod.
This module also displays the following text on the screen:

The number of feet represented by one inch:

GetDimen: The GetDimen module allows the user to choose to plot in
two dimensions or three dimensions.
INPUT: The user is prompted to enter the number of dimensions to
plot. This number must be either two or three.
PROCESSING: If the input operation is unsuccessful or the number
entered is not valid (not equal to two or three), the user is
prompted until a valid response is entered.

OUTPUT: GetDimen returns an integer variable representing the number of dimensions to plot to MainMod. The value of the variable will be either two or three. This module also displays the following text on the screen:

The number of dimensions to plot (either 2 or 3):

GetFileName: This module allows the user to specify the file to use for saving the collected information, then opens the file in the appropriate mode.

INPUT: The user is prompted to enter a one to eight character file name.

PROCESSING: If the input operation is not successful, or the name entered by the user is not between one and eight characters in length, the user is prompted until a valid response is entered. Next, an extension of ".txt" is appended to the name. This name is then appended to the current directory to form a complete file name. If the file exists, the user is prompted to overwrite the file, append the data to the file, or choose a different file name. Processing proceeds according to the users choice. If the choice is to overwrite or append the data, the file is opened in overwrite or append mode, respectively. If the choice is to choose a different file name, the user is again prompted to enter a one to eight character file name. If the file does not exist, a message specifying the full path and file name is displayed, and the file is opened in new mode (i.e. it is created).

OUTPUT: GetFileName modifies a global file variable, assigning a value to the handle of the output file. This module also displays the following text on the screen:

The name of the file to use for saving the point data. The file will be located in the current directory. Enter up to 8 characters, an extension of .txt will be appended by the system:

If the file specified by the user already exists, the following prompt is displayed:

File "drive:\path\name.txt" already exists. Enter 0 to overwrite
the file, A to append the data to the file, or D to enter a different file name:
If the file does not exist, the following is displayed:
The data will be written to file "drive:\path\name.txt"

XYStartPrompt: This module displays a prompt to begin plotting the X and Y coordinates for the current map.
INPUT: There is no input for this module.
PROCESSING: A prompt explaining the procedure for measuring X and Y is displayed and the window used to display point coordinates is defined.
OUTPUT: The following text is displayed on the screen:
Position the cross-hairs over the current point and press button 1 (yellow). Press button 1 again to proceed to the next point, button 2 (white) to remeasure the current point, or button 3 (blue) to quit.
Point X Y

MeasureXY: This module obtains the X and Y coordinates of a point.
INPUT: When a cursor button is pressed, the tablet issues a five byte report. This report, which is received one byte at a time by calling the RecByte module, is the input for MeasureXY. In addition, MainMod passes the integer variable representing the number of the current point and the floating point variable representing the number of feet per line to MeasureXY.
PROCESSING: First, the number of the point to be plotted is displayed. Next, ClearCom is called to clear any bytes from the serial port. The first byte of the five byte report issued by the tablet is the status byte. GetXY checks the status byte to insure that the proximity bit is off and that button one was pressed. If not, a beep is sounded, ClearCom is called to clear the remaining four bytes, and the entire process is repeated. Bytes two and three of the report contain the X position of the cursor, while bytes four and five contain the Y position of the cursor. GetXY converts these packed binary coordinates into decimal values representing
the X and Y coordinates for the current point, then multiplies them by the scale adjustment factor of the current map to make the coordinates true to scale.

OUTPUT: MeasureXY displays the number of the current point in the appropriate column as defined by XYStartPrompt. MeasureXY also returns a floating point structure representing the X and Y coordinates of the current point to MainMod.

StoreXY: This module allows the user to accept or remeasure the coordinates for the current point or quit plotting coordinates. StoreXY also saves the X and Y coordinates of the current point.

INPUT: Input for StoreXY consists of the first byte of the five byte report issued by the tablet when a cursor button is pressed. In addition, MainMod passes a floating point structure representing the X and Y coordinates of the current point and an integer variable representing the number of the current point to the module.

PROCESSING: StoreXY first displays the coordinates for the current point, then waits for the user to press a cursor button. Processing continues according to the button pressed by the user as follows:

1) Button one allows the user to plot the X and Y coordinates for the next point. If this button is pressed, the X and Y coordinates of the current point are saved in the array used to store point coordinates and the number of the current point is incremented by one. If incrementing the number of the current point causes it to be greater than the maximum number of points, a message is displayed and a flag is set to quit plotting coordinates.

2) Button two allows the user to remeasure the X and Y coordinates for the current point. If this button is pressed, the X and Y coordinates for the current point are discarded and the number of the current point is not incremented.

3) Button three allows the user to quit plotting X and Y coordinates. If button three is pressed, the user is prompted to press it again to confirm their choice. If it is pressed
again, the X and Y coordinates of the current point are saved and a flag is set to quit plotting X and Y coordinates. If any other button is pressed, the prompt is cleared and the module waits for the user to press another button.

4) Button four has no function associated with it. If it is pressed, a beep is sounded and the module waits for the user to press another button.

**OUTPUT:** StoreXY passes the following to MainMod:

1) The floating point array used to store point coordinates. If the user chooses to proceed to the next point, the array will contain the X and Y coordinates of the current point.

2) An integer variable representing the updated number of the current point. If the user chooses to proceed to the next point, the number will be incremented by one.

StoreXY also display the X and Y values of the current point in the appropriate column as defined by XYStartPrompt. If the user chooses to proceed to the next point (button one is pressed), and the number of the next point is greater than the maximum number of points allowed per map, the following prompt is displayed:

**The maximum number of points has been reached, terminating plotting.**

If the user chooses to remeasure the point (button two is pressed), the X and Y coordinates of the current point are cleared from the screen. If the user chooses to quit plotting (button three is pressed), the following prompt is displayed:

<<< Press 3 again to confirm >>>

If any button other than three is then pressed, the prompt is cleared.

**ZStartPrompt:** This module displays a prompt to begin entering the Z coordinates for the current map.

**INPUT:** There is no input for this module.

**PROCESSING:** A prompt explaining the procedure for entering the Z coordinates is displayed and the window used to display the coordinates is defined.
OUTPUT: The following text is displayed on the screen:
Type the Z coordinate for the current point. Press enter to proceed
to the next point.
Point X Y Z

GetZ: The GetZ module obtains the Z coordinate of the current
point.
INPUT: Input for GetZ consists of the Z coordinate of the current
point. This number is entered by the user and can be any valid
floating point number. Precision will be rounded to the hundredths
position. In addition, a floating point structure representing the
X and Y coordinates of the current point and an integer variable
representing the number of the current point are passed to the
module by MainMod.
PROCESSING: The number, X value, and Y value of the current point
are displayed. The user must then enter the Z value for the current
point. If the input operation is unsuccessful, the input field is
cleared and the value entered by the user is discarded. This
process continues until a valid number is entered.
OUTPUT: This module returns a floating point structure representing
the coordinates of the current point, updated to contain the Z
coordinate, to MainMod. In addition, the number, X value and Y
value of the current point are displayed in the appropriate column
as defined by ZStartPrompt. The Z value entered by the user is also
echoed to the appropriate column.

StoreZ: This module saves the Z coordinate of the current point.
INPUT: A floating point structure representing the coordinates of
the current point, the floating point array used to store point
coordinates, and an integer variable representing the number of the
current point are passed to StoreZ by MainMod.
PROCESSING: The Z coordinate of the current point is saved in the
appropriate position of the array used to store point coordinates
and the number of the current point is incremented by one.
OUTPUT: StoreZ returns the following to MainMod:
1) The floating point array used to store point measurements, updated to contain the Z coordinate of the current point.
2) An integer variable representing the updated number of the current point.

ZEndPrompt: ZEndPrompt displays a prompt that all Z coordinates have been entered.
INPUT: The user is prompted to press enter.
PROCESSING: ZEndPrompt displays a prompt, then waits for the user to hit enter.
OUTPUT: The following text is displayed on the screen:
   <<< Z has been entered for all points, hit enter to continue >>>

SaveToFile: This module saves the collected point data to a file.
INPUT: MainMod passes a two-dimensional, numeric array containing the coordinates for the points plotted and an integer variable representing the number of points plotted to the module. SaveToFile also accesses a global variable representing the handle of the output file.
PROCESSING: Rows one through n (the number of points) from the array passed by MainMod are written to the specified file in the format described under File Descriptions.
OUTPUT: SaveToFile creates or modifies an ASCII text file (see File Descriptions for format details).

EndPrompt: This module prompts the user to choose to plot another map or quit.
INPUT: The user is prompted to enter a key.
PROCESSING: The user is prompted to enter Y to plot another map or any other key to return to DOS. If the user chooses to plot another map, a boolean variable is set to true. If the user chooses to return to DOS, the boolean variable is set to false.
OUTPUT: This module returns a boolean variable to MainMod, the value of which determines if the program continues or if control returns to DOS.
ClearCom: This module clears any incoming bytes from the serial port.
INPUT: Input for this module consists of zero or more bytes received through the serial port.
PROCESSING: ClearCom checks the status of the serial port a finite number of times. If a byte becomes available it is received. This process is repeated until the status of the port is checked the specified number of times and no byte becomes available.
OUTPUT: There is no output for this module.

ChkTab: This module determines if the tablet is transmitting data.
INPUT: One byte is received from the graphics tablet.
PROCESSING: ChkTab checks the status of the serial port a finite number of times. If a byte becomes available, a boolean flag is set to true. If no byte becomes available, a boolean flag is set to false.
OUTPUT: A boolean value is returned to MainMod.

SendByte: The SendByte module sends a byte through the serial port.
INPUT: A single command byte passed by InitCom.
PROCESSING: The status of the port is checked a finite number of times until the port is ready to transmit a byte. The module then sends the byte to the graphics tablet and returns control to the calling program. If the port does not become available for transmission within the specified time, no data is transmitted.
OUTPUT: One byte is transmitted to the graphics tablet.

RecByte: The RecByte module receives a byte through the serial port.
INPUT: One byte is received from the graphics tablet.
PROCESSING: The status of the port is checked a finite number of times until an incoming character is available, at which time the byte is received. If no byte becomes available in the specified time, a DOS system call is made. This call provides a check for
control-break. If control-break has been entered by the user, the program will terminate, otherwise the entire process is repeated. OUTPUT: RecByte returns the byte received to GetXY.
SUMMARY

After extensive testing by myself, Dr. Berkebile, and other members of the geology department, this system has proved successful in both functionality and usability (See Appendix C, Test Results). In addition to fulfilling all requirements outlined in the proposal, the final system incorporates several enhancements to the original design as outlined in the project proposal. These changes were made after preliminary testing of a prototype of the system. The following list describes the more significant of these enhancements:

1) The sequence in which point coordinates are obtained was modified. Originally, the user was required to measure the X and Y coordinates for the current point using the tablet cursor, then to enter the Z coordinate for the point from the keyboard. This approach resulted in the user being forced to move between the tablet and the keyboard for every point plotted. The current approach requires the user to measure the X and Y coordinates for all points, then to enter the Z coordinates for all points. This reduces movement between the tablet and the keyboard to once per map.

2) Additional functions were assigned to the tablet cursor. The proposed approach utilized only cursor button one, which was used to measure the X and Y coordinates for a point. This function has been maintained, but now after measuring a point the user can press button one again to proceed to the next point, button two to remeasure the current point, or button three to quit plotting.

3) The proposed project required the user to enter the number of points to be plotted, whereas the current system allows the user to measure an unspecified number of points, up to the maximum number of points allowed per map (500).

Possible future enhancements to the system include the following:
1) Integration of the system with the Surfer graphics application, resulting in a single, more powerful system.

2) The displaying of point coordinates not only as numeric data but also as relative points on some portion of the screen (i.e. display the coordinates graphically).

3) Integration of a locally-written or purchased editor with the system to allow the user to modify point coordinates in a text-based form.

4) Modification of the user interface to include menus. The system as it exists now lends itself to a sequential execution of operations, without allowing or requiring the user to modify the standard sequence of events. Enhancements to the system, specifically integration with Surfer or an editor, may require more flexibility with regard to the sequence of operations.
BIBLIOGRAPHY


APPENDIX A
/* INCLUDE FILES **********************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <bios.h>

/* GLOBAL CONSTANTS AND STRUCTURES ******************************************/
#define Com1 0x3f8 /* Address of serial port */
#define ComParams (0x3d0|0x00|0x08|0x03) /* 9600, odd, 8, 1 */
#define RecReady 0x01 /* Receive register ready */
#define RecReg 0 /* Address of receive register */
#define SendReady 0x20 /* Send register ready */
#define SendReg 0 /* Address of send register */
#define StatReg 5 /* Address of status register */
#define Button1 0x01 /* Used to determine which cursor */
#define Button2 0x02 /* button was pressed */
#define Button3 0x03 /* " " " " */
#define ButtonPressed 0x07 /* " " " " */
#define PointMode 0x42 /* Tablet command */
#define ReportRate 0x54 /* " " */
#define ResetTablet 0x00 /* " " */
#define TriggerTablet 0x50 /* " " */
#define LinesPerInch 500.0 /* " " */
#define OutOfProx 0x40 /* Chk stat byte for prox error */
#define MaxPts 500 /* Max number of points to plot */
#define False 0 /* Boolean constant */
#define True !False /* " " */

typedef struct {
  /* Structure for the coordinates of a point */
  float X;
  float Y;
  float Z;
} Coordinates;

/* MODULE DECLARATIONS ******************************************************/
void InitCom();
void StartPrompt();
void GetScale();
void GetDimen();
void GetName();
void XYStartPrompt();
void MeasureXY();
void StoreXY();
void ZStartPrompt();
void GetZ();
void StoreZ();
void ZEndPrompt();
void SaveToFile();
void EndPrompt();
int ChkTab();
void ClearCom();
void SendByte();
int RecByte();

FILE *FileStream; /* Global handle for output file */

/******************************************************************************/
main ()
/* This is the controlling module for the system. It performs the */
/* necessary module calls and variable passing for the system to */
/* execute. */
{
	int Continue, /* Flag to control main program loop */
	XYDone, /* Flag to control plotting X,Y loop */
	NumPts, /* Number of points to be plotted */
	PtNum, /* Number of current point */
	Dimen; /* Number of dimensions be plotted */
	float FeetPerLine, /* Scale of map in tablet units */
	PtArray[MaxPts][3]; /* All coordinates for current map */
	Coordinates Point; /* Coordinates of current point */

InitCom(&Continue);

while (Continue) {
	StartPrompt();
	GetScale(&FeetPerLine);
	GetDimen(&Dimen);
	GetFileName();

	XYStartPrompt();
	PtNum=1;
	XYDone = False;
	while (!XYDone) {
		MeasureXY(&Point, PtNum, &FeetPerLine);
		StoreXY(&Point, PtArray, &PtNum, &XYDone);
	}
	NumPts = PtNum;
	if (Dimen == 3) {
		ZStartPrompt();
		PtNum = 1;
		while (PtNum <= NumPts) {
			GetZ(&Point, PtArray, PtNum);
			StoreZ(&Point, PtArray, &PtNum);
		}
		ZEndPrompt();
	}
	SaveToFile(PtArray, NumPts, Dimen);
	EndPrompt(&Continue);
}
clrscr();

/******************************************************************************/
void InitCom(int *Continue)
/* This module configures Com1, then resets and initializes the */
/* graphics tablet. InitCom then calls ChkTab in insure that the */
/* tablet is responding. */
{
	int TabReady;
	char Resp;

	bioscom(0,ComParams,0); /* Set Com1 to proper setting */
	SendByte(ResetTablet); /* Send commands to configure tablet */
sleep();
SendByte(PointMode);
SendByte(ReportRate);

ClearCom();
SendByte(TriggerTablet); /* Send trigger command. If tablet does */
TabReady = ChkTab();    /* not respond, set flag to return to DOS */
if (!TabReady)
{
    printf("   << Tablet not responding, returning to DOS. >>\n");
    *Continue = False;
}
else
    *Continue = True;
return;
}

//-----------------------------------------------
void StartPrompt()
/* This module displays the initial program prompt    */
{

clrscr();
setcbk(1);      /* Set control break on      */
printf("   <<< Press Ctrl-Break at any point to return to DOS. >>\n\n");
printf("\n   Please enter the following parameters for the current map.\n\n");
return;
}

//-----------------------------------------------
void GetScale(float *FeetPerLine)
/* This module prompts the user to enter the scale for the current */
/* map (FeetPerInch) until a valid number is entered. It then */
/* calculates the scale to use for the current map (FeetPerLine). */
{
    float   FeetPerInch;
    int     GoodScan;

    GoodScan = False;
    while (!GoodScan) {
        printf("  The number of feet represented by one inch: ");
        fflush(stdin);
        GoodScan = scanf(" %f",&FeetPerInch);
    }
    *FeetPerLine = (FeetPerInch / LinesPerInch);

    return;
}

//-----------------------------------------------
void GetDimen(int *Dimen)
/* This module prompts the user to enter the dimensions for the */
/* current map until either 2 or 3 is entered.                      */
{
    int    GoodScan;

    GoodScan = False;
    *Dimen = 0;
    while (!GoodScan) {
        printf("  The number of dimensions to plot (either 2 or 3): ");
        fflush(stdin);
        GoodScan = scanf(" %d",Dimen);
    }

    return;
}

//-----------------------------------------------
void GetFileName()
int i, NameLength, PathLength, GoodName, GoodResp, GoodScan;
char Resp, FileName[31], FilePath[31];

GoodName = False;

while (!GoodName) {
    printf("\n The name of the file to use for saving the point data. The file\n");
    printf(" will be located in the current directory. Enter up to 8 characters,\n");
    printf(" an extension of .txt will be appended by the system: ");
    fflush(stdin);
    GoodScan = scanf("[0-9A-Za-z]",FileName);
    if (GoodScan && (strlen(FileName) >= 1 && strlen(FileName) <= 8)) {
        GoodName = True;
        charcmd(FilePath,30);
        strcat(FilePath,"\"\") ; /* Combine current directory and */
        strcat(FileName,".txt"); /* name entered by user to make */
        NameLength = strlen(FileName); /* complete file name {ends up */
        PathLength = strlen(FilePath); /* in FilePath */
        for (i=0;i<=NameLength;+i)
            FilePath[PathLength+i] = toupper(FileName[i]);
        if (searchpath(FilePath)!=NULL) { /* Check if file exists */
            GoodResp = False;
            while (!GoodResp) {
                GoodResp = True;
                printf("\n File %s already exists.\n", FilePath);
                printf(" Enter O to overwrite the file, A to append the data to the file,\n");
                printf(" or D to enter a different file name: ");
                fflush(stdin);
                scanf("%c",&Resp); /* Open file according to Resp */
                switch (toupper(Resp)) {
                case 'O' : FileStream = fopen(FilePath,"w+t"); break;
                case 'A' : FileStream = fopen(FilePath,"a+t"); break;
                case 'D' : GoodName = False; break;
                default : GoodResp = False;
                }
            }
        } else { /* If file does not exist, create it */
            printf("\n The data will be written to file %s \n", FilePath);
            FileStream = fopen(FilePath,"w+t");
            sleep(2);
        }
    }
}

return;
}

/*******************************************************************************/
void XYStartPrompt()
/*****************************************************************************/
/* This module displays the screen heading for plotting X and Y, */
/* than defines the window used to display measurements. */
*******************************************************************************/
clrsr();
printf("\n Position the cross-hairs over the current point and press button 1\n");
sprintf(" (yellow). Press button 1 again to proceed to the next point, button 2\n");
printf(" (white) to remeasure the current point, or button 3 (blue) to quit.\n");
printf(" Point X Y\n");
window(wherex(),wherey(),80,23); /* Define data window */
clrsr();
return;
}

/*****************************/

void MeasureXY(Coordinates *Point, int PtNum, float *FeetPerLine)
/* This module prompts the user to measure the current point (using */
/* the cursor) then receives the status and measurement bytes using */
/* the RecByte module. */
{
    int Byte1, Byte2, Byte3, Byte4, Byte5, i, GoodButton;

cprintf("%d",PtNum);
    GoodButton = False;
    while (!GoodButton) { /* Loop until the user presses 1 and the */
        ClearCom(); /* cursor is in proximity */
        Byte1 = RecByte();
        if (((Byte1 & ButtonPressed) == Button1) && ((Byte1 & OutOfProx)))
            GoodButton = True;
        else
            printf("\a");
    }
    Byte2 = RecByte(); /* Receive data bytes */
    Byte3 = RecByte();
    Byte4 = RecByte();
    Byte5 = RecByte();

    Point-&gt;X = (Byte3 * 128 + Byte2) * FeetPerLine; /* Calculate X,Y and */
    Point-&gt;Y = (Byte5 * 128 + Byte4) * FeetPerLine; /* convert to scale */
    return;
}

/*****************************/

void StoreXY(Coordinates *Point, float PtArray[][3], int *PtNum, int *XYDone)
/* This module displays X and Y for the current point then waits for */
/* a cursor button to be pressed. Processing continues as follows: */
/* Button 3 pressed: Prompt to press again. If pressed, exit loop. */
/* Button 2 pressed: Remeasure current point (don't increment PtNum). */
/* Button 1 pressed: Save current measurements and increment PtNum. */
{
    int Byte1, EndX, GoodButton;

cprintf("%12.2f %12.2f",Point-&gt;X,Point-&gt;Y);
    EndX = where(); /* Save position of input field */

    GoodButton = False;
    while (!GoodButton) { /* Loop until 1,2, or 3 is pressed */
        ClearCom();
        Byte1 = RecByte();
        /* If 3 pressed, prompt */
        if (((Byte1 & ButtonPressed) == Button3) { /* to press again */
            printf("\a");
            cprintf("<<< Press 3 again to confirm >>>");
            ClearCom();
            Byte1 = RecByte();
            if (((Byte1 & ButtonPressed) == Button3) {
                GoodButton = True;
                *XYDone = True;
                PtArray[*PtNum][1] = Point-&gt;X; /* If 3 again, save X,Y, set*/
                PtArray[*PtNum][2] = Point-&gt;Y; /* flag to exit and flag to */
            } /* quit plotting X and Y */
            else {
                gotoxy(EndX,wherey());
                cleat();
            }
        }
        /* If 2 pressed, */
    }
}
if (Byte1 & ButtonPressed) == Button2 { /* remeasure point */
    GoodButton = True;
gotoxy(1,wherey()); /* Clear X,Y and */
circol(); /* and continue */
}
else /* If 1 pressed, */
if ((Byte1 & ButtonPressed) == Button1) { /* proceed to */
    GoodButton = True; /* next point */
cprintf("\\n");
gotoxy(1,wherey()); /* Save X,Y, and */
PtArray[PtNum][1] = Point->X; /* increment PtNum */
PtArray[PtNum][2] = Point->Y;
++PtNum;
if (*PtNum > MaxPts) { /* If MaxPts exceeded */
    *PtNum = MaxPts; /* quit plotting */
    printf("\a\a\n");
cprintf("The maximum number of points has been reached, terminating plotting."),
sleep(3);
    *XTDone = True;
}
else /* If 4 pressed, beep and */
    printf("\a\a\n"); /* wait for next button */
}
return;

*******************************************************************************
void ZStartPrompt()
/* This module displays the screen heading for entering Z values then*/
/* defines the window used to display X and Y and enter Z. */
{
    window(1,1,80,23); /* Define window as full fullscreen */
circol();
    printf("\n Type the Z coordinate for the current point. Press enter to proceed to the\n");
    printf("next point.\n\n\nPoint X Y Z\n");
    window(wherex(),wherey(),80,20); /* Define data window */
circol();
return;

*******************************************************************************
void GetZ(Coordinates *Point, float PtArray[])[3], int PtNum)
/* This module displays the number and X and Y measurements for the */
/* current point then waits for the user to enter a valid Z value. */
{
    int Y, GoodScan;
    Y = wherey(); /* Save current line position */
    GoodScan = False;
    while (!{GoodScan}) {
        cprintf("%d\%11.2f\%11.2f\",PtNum, PtArray[PtNum][1], PtArray[PtNum][2]);
        fflush(stdin);
        GoodScan = scanf("%f",&Point->Z); /* scanf scans to the next */
        gotoxy(1,Y); /* line so jump back 1 line */
        circol();
    }
    cprintf("%d\%11.2f\%11.2f\%11.2f\n",PtNum, PtArray[PtNum][1], PtArray[PtNum][2], Point->Z);
    gotoxy(1, wherey()); /* goto first column of new line */
    return;
}
*******************************************************************************
void Store2(Coordinates *Point, float PtArray[])[3], int *PtNum)
/* This module saves 2 for the current point and increments PtNum. */
PtArray[*PtNum][3] = Point->Z;
+++PtNum;
return;
}

/** This module displays a prompt that all Z measurements have been entered then waits for the user to hit enter. */

int GoodScan;

printf("\n
<< Z has been entered for all points, hit enter to continue >>\n");
fflush(stdin);
scanf("%c",&GoodScan); /* scan used to detect when enter is hit, */
return; /* no assignment is made */

/** This module save the collected point measurements for the current map to the file specified by the user in the GetFileName module. */

if (Dimen == 3) /* write X,Y,Z */
    for (i = 1; i <= NumPts; ++i)
        fprintf(FileStream, "%11.2f %11.2f %11.2f \n", PtArray[i][1], PtArray[i][2], PtArray[i][3]);
else /* if 2-D write X,Y */
    for (i = 1; i <= NumPts; ++i)
        fprintf(FileStream, "%11.2f %11.2f \n", PtArray[i][1], PtArray[i][2]);
fflush(FileStream);
fclose(FileStream);
return;

/** This module prompts the user to choose to plot another map or quit */

window(1,1,80,25); /* Define window as full fullscreen */
clrscr();
printf("\n\n Enter Y if you wish to plot another map: ");
fflush(stdin);
scanf("%c",&Response); /* If user enters Y, continue. If */
if(toupper(Response) == 'Y') /* user enters any other key or */
    *Continue = True; /* just hits enter, quit. */
else
    *Continue = False;
return;

/** This module clears any incoming bytes from Com1. */

int Count, ComClear;
ComClear = False; /* Set flag to get in loop */
Counter = 0;
while (!ComClear) {
    /* Count until Com1 stat indicates byte waiting or count exceeded */
while (!inportb(Com1 + StatReg) & RecReady) & Counter < 30000) {Counter++;

if (Counter >= 30000) /* If no byte became available before the */
    ComClear = True; /* counter reached 30000, the port is */
else /* clear. Otherwise receive the byte and */
    inportb(Com1); /* get back in the loop to wait for the */
    /* next byte. */
return;
}

/*******************************************************************************/
int ChkTab()
/* This module returns true if the tablet is transmitting or false */
/* if it is not. */
{
int Counter, TabReady;

Counter = 0;

/* Count until Com1 stat indicates byte waiting or count exceeded */
while (!((inportb(Com1 + StatReg) & RecReady) & Counter < 30000) {Counter++;

if (Counter >= 30000) /* If no byte became available before the */
    TabReady = False; /* counter reached 30000, the tablet is */
else /* not transmitting, return False. */
    TabReady = True; /* Otherwise the tablet is transmitting, */
return(TabReady); /* return True. */
}

/*******************************************************************************/
void SendByte(char Ch)
/* This module sends a byte through Com1. */
{
int Counter;

Counter = 0;

/* Count until Com1 stat indicates ready to send or count exceeded */
while(!((inportb(Com1 + StatReg) & SendReady) & Counter < 30000) {Counter++;

if (Counter < 30000) /* If the port became available, */
    outportb(Com1, Ch); /* send the char out Com1 */
return;
}

/*******************************************************************************/
int RecByte4()
/* This module receives a byte through Com1. */
{
long Counter;
int ByteRecd;

ByteRecd=False;

while (!ByteRecd) {
    Counter = 0;

    /* Count until Com1 stat indicates byte waiting or count exceeded */
    while (!((inportb(Com1 + StatReg) & RecReady) & Counter < 5000000) {Counter++;

    if (Counter >= 500000) /* If no byte arrived before the counter */
        bdi(0x19,0,0); /* reached 500000, make a system call to */
else /* check for control-break, else continue */
    ByteRecd=True; /* to wait for a byte. */
}
return(inportb(Com1)); /* Return the received byte. */
}

/*******************************************************************************/
APPENDIX B
<table>
<thead>
<tr>
<th>Point Number</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7640</td>
<td>2360</td>
<td>422</td>
</tr>
<tr>
<td>2</td>
<td>7330</td>
<td>5740</td>
<td>486</td>
</tr>
<tr>
<td>3</td>
<td>6700</td>
<td>1060</td>
<td>452</td>
</tr>
<tr>
<td>4</td>
<td>6740</td>
<td>2060</td>
<td>458</td>
</tr>
<tr>
<td>5</td>
<td>6900</td>
<td>2640</td>
<td>396</td>
</tr>
<tr>
<td>6</td>
<td>6490</td>
<td>4600</td>
<td>444</td>
</tr>
<tr>
<td>7</td>
<td>5750</td>
<td>560</td>
<td>438</td>
</tr>
<tr>
<td>8</td>
<td>5700</td>
<td>2000</td>
<td>445</td>
</tr>
<tr>
<td>9</td>
<td>5550</td>
<td>3960</td>
<td>402</td>
</tr>
<tr>
<td>10</td>
<td>5230</td>
<td>1810</td>
<td>435</td>
</tr>
<tr>
<td>11</td>
<td>5110</td>
<td>2750</td>
<td>457</td>
</tr>
<tr>
<td>12</td>
<td>4470</td>
<td>400</td>
<td>550</td>
</tr>
<tr>
<td>13</td>
<td>4350</td>
<td>2450</td>
<td>502</td>
</tr>
<tr>
<td>14</td>
<td>4600</td>
<td>4790</td>
<td>497</td>
</tr>
<tr>
<td>15</td>
<td>4080</td>
<td>3940</td>
<td>550</td>
</tr>
<tr>
<td>16</td>
<td>3590</td>
<td>1190</td>
<td>550</td>
</tr>
<tr>
<td>17</td>
<td>3700</td>
<td>5000</td>
<td>558</td>
</tr>
<tr>
<td>18</td>
<td>3000</td>
<td>2250</td>
<td>508</td>
</tr>
<tr>
<td>19</td>
<td>3000</td>
<td>3050</td>
<td>468</td>
</tr>
<tr>
<td>20</td>
<td>3000</td>
<td>4100</td>
<td>552</td>
</tr>
<tr>
<td>21</td>
<td>2600</td>
<td>3880</td>
<td>523</td>
</tr>
<tr>
<td>22</td>
<td>2460</td>
<td>5710</td>
<td>551</td>
</tr>
<tr>
<td>23</td>
<td>2210</td>
<td>770</td>
<td>560</td>
</tr>
<tr>
<td>24</td>
<td>2180</td>
<td>4780</td>
<td>555</td>
</tr>
<tr>
<td>25</td>
<td>1750</td>
<td>540</td>
<td>549</td>
</tr>
<tr>
<td>26</td>
<td>1860</td>
<td>2350</td>
<td>558</td>
</tr>
<tr>
<td>27</td>
<td>1500</td>
<td>4120</td>
<td>537</td>
</tr>
<tr>
<td>28</td>
<td>1050</td>
<td>3000</td>
<td>562</td>
</tr>
<tr>
<td>29</td>
<td>750</td>
<td>1750</td>
<td>550</td>
</tr>
<tr>
<td>30</td>
<td>900</td>
<td>3860</td>
<td>550</td>
</tr>
<tr>
<td>31</td>
<td>800</td>
<td>5380</td>
<td>576</td>
</tr>
<tr>
<td>32</td>
<td>450</td>
<td>380</td>
<td>582</td>
</tr>
<tr>
<td>Point Number</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>1</td>
<td>7420.00</td>
<td>2374.00</td>
<td>422.00</td>
</tr>
<tr>
<td>2</td>
<td>7080.00</td>
<td>5672.00</td>
<td>486.00</td>
</tr>
<tr>
<td>3</td>
<td>6506.00</td>
<td>1076.00</td>
<td>452.00</td>
</tr>
<tr>
<td>4</td>
<td>6518.00</td>
<td>2068.00</td>
<td>458.00</td>
</tr>
<tr>
<td>5</td>
<td>6674.00</td>
<td>2632.00</td>
<td>396.00</td>
</tr>
<tr>
<td>6</td>
<td>6240.00</td>
<td>4572.00</td>
<td>444.00</td>
</tr>
<tr>
<td>7</td>
<td>5552.00</td>
<td>576.00</td>
<td>438.00</td>
</tr>
<tr>
<td>8</td>
<td>5490.00</td>
<td>2004.00</td>
<td>445.00</td>
</tr>
<tr>
<td>9</td>
<td>5300.00</td>
<td>3940.00</td>
<td>402.00</td>
</tr>
<tr>
<td>10</td>
<td>5010.00</td>
<td>1818.00</td>
<td>435.00</td>
</tr>
<tr>
<td>11</td>
<td>4896.00</td>
<td>2742.00</td>
<td>457.00</td>
</tr>
<tr>
<td>12</td>
<td>4272.00</td>
<td>420.00</td>
<td>550.00</td>
</tr>
<tr>
<td>13</td>
<td>4134.00</td>
<td>2448.00</td>
<td>502.00</td>
</tr>
<tr>
<td>14</td>
<td>4340.00</td>
<td>4736.00</td>
<td>497.00</td>
</tr>
<tr>
<td>15</td>
<td>3828.00</td>
<td>3900.00</td>
<td>550.00</td>
</tr>
<tr>
<td>16</td>
<td>3378.00</td>
<td>1176.00</td>
<td>550.00</td>
</tr>
<tr>
<td>17</td>
<td>3420.00</td>
<td>4958.00</td>
<td>558.00</td>
</tr>
<tr>
<td>18</td>
<td>2764.00</td>
<td>2238.00</td>
<td>508.00</td>
</tr>
<tr>
<td>19</td>
<td>2758.00</td>
<td>3026.00</td>
<td>468.00</td>
</tr>
<tr>
<td>20</td>
<td>2752.00</td>
<td>4044.00</td>
<td>552.00</td>
</tr>
<tr>
<td>21</td>
<td>2348.00</td>
<td>3842.00</td>
<td>523.00</td>
</tr>
<tr>
<td>22</td>
<td>2188.00</td>
<td>5650.00</td>
<td>551.00</td>
</tr>
<tr>
<td>23</td>
<td>1992.00</td>
<td>758.00</td>
<td>560.00</td>
</tr>
<tr>
<td>24</td>
<td>1892.00</td>
<td>4740.00</td>
<td>555.00</td>
</tr>
<tr>
<td>25</td>
<td>1538.00</td>
<td>522.00</td>
<td>549.00</td>
</tr>
<tr>
<td>26</td>
<td>1624.00</td>
<td>2330.00</td>
<td>558.00</td>
</tr>
<tr>
<td>27</td>
<td>1238.00</td>
<td>4076.00</td>
<td>537.00</td>
</tr>
<tr>
<td>28</td>
<td>780.00</td>
<td>2980.00</td>
<td>562.00</td>
</tr>
<tr>
<td>29</td>
<td>502.00</td>
<td>1760.00</td>
<td>550.00</td>
</tr>
<tr>
<td>30</td>
<td>636.00</td>
<td>3822.00</td>
<td>550.00</td>
</tr>
<tr>
<td>31</td>
<td>516.00</td>
<td>5300.00</td>
<td>576.00</td>
</tr>
<tr>
<td>32</td>
<td>218.00</td>
<td>378.00</td>
<td>582.00</td>
</tr>
<tr>
<td>COORDINATES MEASURED BY HAND</td>
<td>COORDINATES MEASURED USING SYSTEM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n )</td>
<td>( X(n) - X(n-1) )</td>
<td>( Y(n) - Y(n-1) )</td>
<td>( n )</td>
</tr>
<tr>
<td>2</td>
<td>-310</td>
<td>+3380</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>-630</td>
<td>-4680</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>+40</td>
<td>+1000</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>+160</td>
<td>+580</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>-410</td>
<td>+1960</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>-740</td>
<td>-4040</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>-50</td>
<td>+1440</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>-150</td>
<td>+1960</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>-320</td>
<td>-2150</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>-120</td>
<td>+940</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>-640</td>
<td>-2350</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>-120</td>
<td>+2050</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>+250</td>
<td>+2340</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>-520</td>
<td>-850</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>-490</td>
<td>-2750</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>+110</td>
<td>+3810</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>-700</td>
<td>-2750</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>+800</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>+1050</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>-400</td>
<td>-220</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>-140</td>
<td>+1830</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>-250</td>
<td>-4940</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>-30</td>
<td>+4010</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>-430</td>
<td>-4240</td>
<td>25</td>
</tr>
<tr>
<td>26</td>
<td>+110</td>
<td>+1810</td>
<td>26</td>
</tr>
<tr>
<td>27</td>
<td>-360</td>
<td>+1770</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>-450</td>
<td>-1120</td>
<td>28</td>
</tr>
<tr>
<td>29</td>
<td>-300</td>
<td>-1250</td>
<td>29</td>
</tr>
<tr>
<td>30</td>
<td>+150</td>
<td>+2110</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>-100</td>
<td>+1520</td>
<td>31</td>
</tr>
<tr>
<td>32</td>
<td>-350</td>
<td>-5000</td>
<td>32</td>
</tr>
</tbody>
</table>
TOPOGRAPHIC MAP PRODUCED FROM POINT COORDINATES MEASURED BY HAND