WANG

2262 X-Y DIGITIZER REFERENCE MANUAL

SYSTEM 2200

2262 X-Y Digitizer

Reference Manual

WANG LABORATORIES, INC.
836 NORTH STREET, TEWKSBURY, MASSACHUSETTS 01876, TEL. (617) 851-4111, TWX 710 343-6769, TELEX 94-7421

Disclaimer of Warranties and Limitation of Liabilities

The staff of Wang Laboratories, Inc., has taken due care in preparing this manual; however, nothing contained herein modifies or alters in any way the standard terms and conditions of the Wang purchase agreement, lease agreement, or rental agreement by which this equipment was acquired, nor increases in any way Wang's liability to the customer. In no event shall Wang Laboratories, Inc., or its subsidiaries be liable for incidental or consequential damages in connection with or arising from the use of this manual or any programs contained herein.

WANG LABORATORIES, INC.
836 NORTH STREET, TEWKSBURY, MASSACHUSETTS 01876, TEL (617) 851-4111, TWX 710 343-6769, TELEX 94-7421

HOW TO USE THIS MANUAL

The Model 2262 X-Y Digitizer Reference Manual is designed to provide instruction in the operation and programming of the 2262 Digitizer. Those chapters which discuss programming techniques (Chapters 4 and 5) assume some knowledge of the System 2200 BASIC language. The general forms of all BASIC statements associated with digitizer operation are included in an Appendix at the back of the manual for quick reference.

TABLE OF CONTENTS

LIST OF EXAMPLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION TO THE MODEL 2262 X-Y DIGITIZER

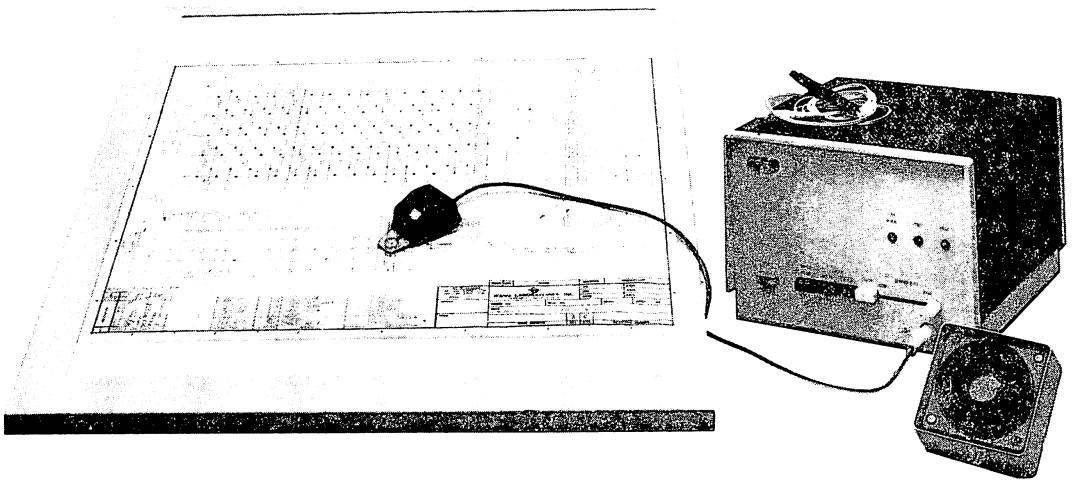

Figure 1-1. Typical Digitizer Configuration

1.1 INTRODUCTION

The Wang Model 2262 X-Y Digitizer is an input device used to locate the positions of points on graphic media such as strip charts, graphs, photos, engineering drawings, x-rays, etc., and convert the points into digital information (in the form of cartesian coordinates) which can be transmitted to the system for storage and analysis. The digitizer generates pairs of X-Y coordinates for discrete points on a document, and transmits them to the system, where they can be analyzed, replotted to scale, or stored off on tape or disk for future processing.

A standard digitizer configuration includes a digitizer tablet, a separate chassis containing the necessary electronics, and a digitizing implement. (Two types of implements, a bull's-eye cursor and a pen stylus, are provided.) The source document must be placed on the digitizing surface of the tablet; the operator then manually traces curvilinear data, or digitizes individual points on the document, with one of the digitizing implements. Although both digitizing implements provide the operator with unencumbered freedom of movement across the digitizing area, they are designed for somewhat different sorts of applications. The cursor is designed for detailed work in which accuracy is critical, while the pen stylus is suited for inking and faster, rougher input.

Different applications also may require different modes of digitizing. The Model 2262 provides three digitizing modes: Single Point, Switch Stream, and $\overline{\text{Switch}}$ Stream. ($\overline{\text{Switch}}$ Stream should be read "Non-Switch" Stream, and implies that a switch is not required to digitize points.) In Single Point mode, the operator can digitize and transmit individual points. This mode of digitizing is useful in many types of graphic analysis and data input, where only selected sample points are

wanted. For operations such as curve tracing, in which a large number of sequential points must be digitized as the line is traced, the two Stream modes enable the operator to digitize and transmit a steady stream of points.

Manual operation of a digitizing implement to digitize points on a document is, of course, only a part of the total digitizing process. The coordinates transmitted by the digitizer must be received and processed in the system under program control. The speed and efficiency with which points are processed is determined principally by the efficiency of the controlling program. A variety of BASIC statements, including INPUT, DATALOAD, DATALOAD BT, and MAT INPUT, are available for reading in digitizer input. Each of these statements has certain characteristics which make it particularly suitable for some applications, and unsuitable for others. Several programming techniques and considerations for processing data from the digitizer are discussed in Chapter 4. (Note that, while the digitizer can be interfaced to any Wang system, not all of the above statements are available on all systems. The System 2200A, for example, does not provide DATALOAD, DATALOAD BT, or MAT INPUT. The System 2200S and WCS/10 do not provide those statements either, unless equipped with one of the Options 22, 23, or 24.)

1.2 MODEL 2262 DIGITIZER COMPONENTS

The Model 2262 X-Y Digitizer includes as standard equipment all of the following components:

- A digitizer tablet, upon which the document to be analyzed is placed. The standard size tablet, provided with the Model 2262-1, is 20 inches x 20 inches (51 cm x 51 cm) in size. Two larger tablets are also available. The Model 2262-2 offers a tablet 30" x 40" (75 cm x 100 cm) in size, while the Model 2262-3 comes equipped with a 36" x 48" (80 cm x 120 cm) tablet.

- A hand-held pen stylus and a four-button cursor, used to locate the points on a document to be digitized, and digitize them. The pen stylus holds a standard ball point pen refill, which can be used to trace a continuous line or to indicate discrete points on a document. The cursor has a black bull's eye sight which is used for the same purpose. Both a pen stylus and a cursor are provided with the digitizer, but only one can be plugged into the digitizer at a given time.

- A digitizer chassis, which holds the power supply and all necessary electronics for the digitizer.

- A Wang Model 2252A Input Interface Controller Board, which serves to interface the Model 2262 Digitizer to the System 2200. A cable connecting the digitizer chassis to the 2252A board in the CPU is also provided.

- A bias magnetic strip for degaussing the tablet should its magnetic bias accidentally be altered by exposure to a strong magnetic field.

Optionally, an audio annunciator is available with the digitizer at a small additional cost. The annunciator consists of a small remote speaker which can be plugged into the rear panel of the digitizer chassis, and located anywhere adjacent to the digitizer tablet. The annunciator produces an audible "beep" each time a point is transmitted from the digitizer to the System 2200, providing the operator with an audible verification of the digitizer operation without requiring him to look away from the work area. The annunciator has a volume control knob which permits adjustment of the loudness of the tone.

1.3 UNPACKING & INSPECTION

Carefully unpack the equipment and inspect all components for damage. Pay special attention to the digitizer tablet, checking it closely for cracks or chips. If there is any indication of damage, notify the shipping agency at once. Check all equipment received against the purchase order. Decals specifying model numbers are affixed to all Wang equipment, usually on the back of the unit.

Note that a flexible magnet is packed with the digitizing implements and the digitizer chassis, while the tablet is packed in a separate carton. The magnet should be unpacked and stored away from the digitizer tablet and from tape cassettes or disk platters. Do not allow the magnet to rest on or near the digitizer tablet, except under the controlled conditions described in Appendix B. The magnet is used to restore the magnetic bias of the tablet, should the bias accidentally be altered (see Appendix B and the Warning below).

> WARNING:
>
> The digitizer tablet has a permanently biased magnetic substrate beneath the vinyl digitizing surface. If magnetic objects or large metal objects are placed on the tablet, or if the tablet is subjected to a high-intensity magnetic field (greater than about 40 gauss), the magnetic bias will be affected. Alteration of the bias may, in turn, affect the accuracy of the digitizer. Proper bias can be restored to the tablet with the procedure described in Appendix B.

1.4 INSTALLATION

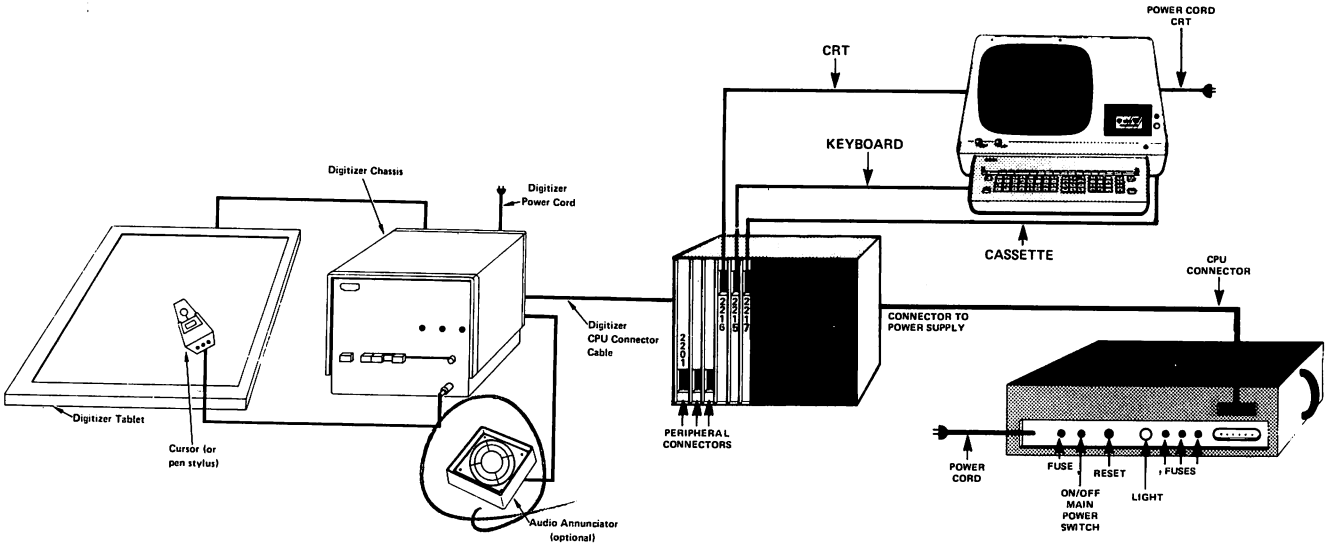

Figure 1-2. Typical System Configuration

When you have unpacked and inspected all of your system components, observe the following installation procedure:

1. Arrange the digitizer chassis, digitizer tablet, audio annunciator (if you have purchased it) and other system components (such as the keyboard and CRT) in the working area. The units must be grouped so that CPU connector cables from all peripherals will reach the appropriate slots in the System 2200 CPU, and power supply cords will reach available sockets.

2. Two of the four margins on the digitizer tablet are somewhat wider than the other two margins. Position the tablet so that the wider margins form the lefthand and upper margins of the tablet, while the narrower margins become the righthand and lower margins (see Figure 1-3). In this position, the tablet's connector cord should extend from beneath the upper edge of the tablet. The tablet is now properly oriented for first quadrant operations, with the absolute origin point located in the lower lefthand corner.

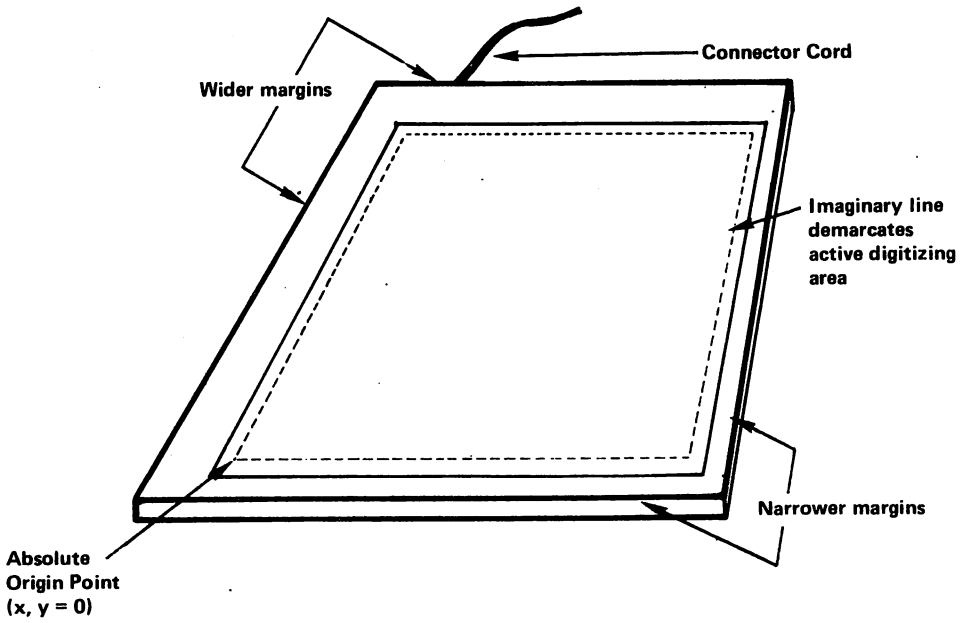

Figure 1-3. Proper Orientation of the Digitizer Tablet

3. Plug the tablet's connector plug into the jack marked "TABLET" on the rear panel of the digitizer chassis. Hold the plug so that the red or silver clip is facing directly upward, and firmly press the plug into the jack.

4. Plug the connector plug of the cursor or stylus into the jack marked "INPUT" on the front panel of the digitizer chassis.

5. (Optional) If you have purchased an audio annunciator, plug it into the unmarked jack on the rear panel of the digitizer chassis. The annunciator may be located anywhere adjacent to the digitizer tablet. A thumbwheel switch on the annunciator permits you to adjust the volume of the audible tone.

6. Plug the CPU connector cable from the rear panel of the digitizer chassis into the female receptacle on the Model 2252A Controller Board in the CPU. (Note: All power must be OFF.) When the plug is attached, make sure that the lock clips are snapped into place at the CPU connection. The Model 2252A board has a series of pushbuttons; be sure that the two pushbuttons closest to the connector (marked "IS" and "SIGN") are DOWN. All other buttons must be UP (see Figure 1-4).

7. Set up all other system peripherals according to procedures prescribed for those peripherals.

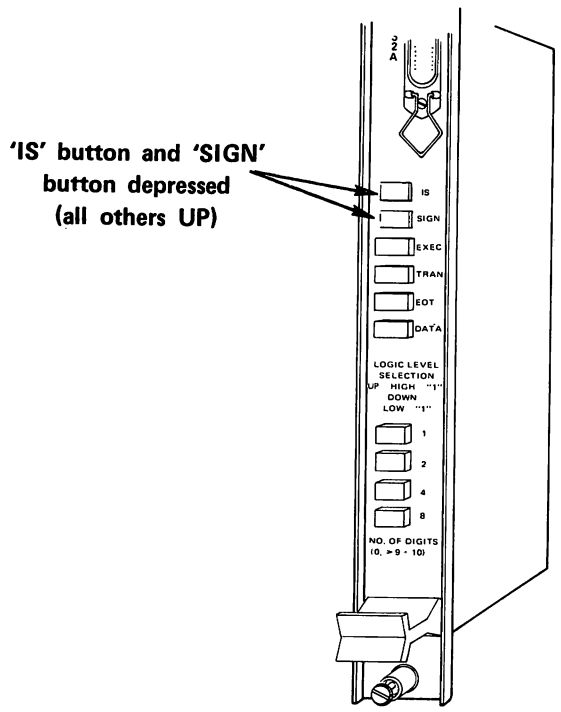

Figure 1-4. The Model 2252A Input Interface Controller Board Set for the Digitizer. The IS and SIGN Buttons are Depressed. All Others Are UP.

1.5 POWER-ON PROCEDURE

1. Switch ON the power switches on all peripherals equipped with them (some peripherals, like the keyboard, do not have power switches; they are powered on automatically when the main power switch is switched on).

2. Depress the POWER button on the front panel of the digitizer chassis.

3. Switch ON the main power toggle switch on the Power Supply Unit. The red light next to the switch glows on, indicating proper system operation. The action of switching on the main power switch Master Initializes the system.

The digitizer is now ready for use. There are, of course, some important points which must be considered before you begin entering data with the digitizer. Chapter 2 examines the features and operation of the four-button cursor and pen stylus, and explains the significances of the various lights and buttons on the front panel of the digitizer chassis. Chapter 3 discusses the operation of the digitizer, explains the format of data points transmitted to the System 2200, and discusses the three modes of digitizer operation. Chapters 4 and 5, finally, discuss the BASIC instructions available on the System 2200 for reading data from the digitizer, and present some important programming considerations.

CHAPTER 2

GENERAL INFORMATION ON THE DIGITIZER

2.1 THE DIGITIZING IMPLEMENTS (FOUR-BUTTON CURSOR AND PEN STYLUS)

Two digitizing elements are provided with the digitizer, a four-button cursor and a pen stylus. The two implements are designed for different types of digitizing operations. Only one can be plugged into the digitizer chassis at a given time.

The Four-Button Cursor

The cursor has a black bull's eye sight, a large square button called the Flag 0 pushbutton, and three smaller pushbuttons called the Flag 1, 2, and 4 pushbuttons (see Figure 2-1). In the normal case, points are digitized in single point or switch stream mode by depressing the Flag 0 pushbutton when the point or line has been aligned in the center of the bull's eye. In this case, the Flag digit for each readout value (see Figure 3-2) is set to zero. Any one of the Flag buttons 1, 2, or 4 can be used to digitized points as well. In this case, the Flag digit in each readout transmitted is set to one, two, or four, depending upon the Flag button used.

The special flag digits set by the Flag buttons can be extremely helpful in identifying particular data points for special conditions, such as indicating initialization and termination conditions to the operating program, identifying the closure point in area calculations, or signalling the end of job; in plotting, flags are often used to signal pen up and pen down conditions. The flag digits are discussed in greater detail in Chapter 3, sections 3.3 and 3.4.

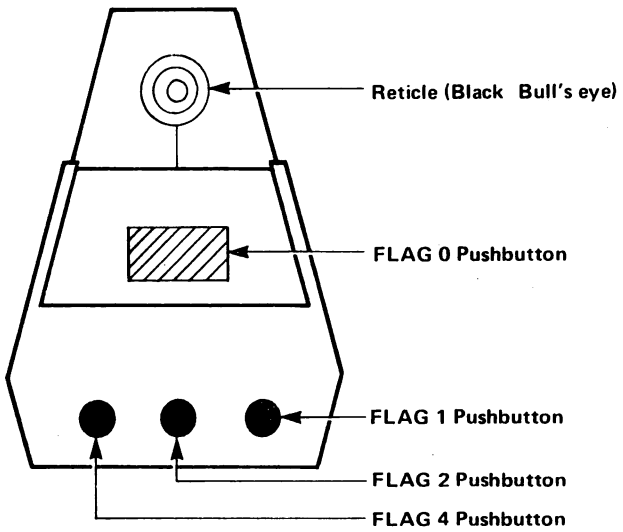

Figure 2-1. The Four-Button Cursor

The Pen Stylus

The pen stylus resembles a normal ballpoint pen in size and shape, and is loaded with a standard ballpoint pen refill. Points are digitized with the pen stylus by positioning the pen tip on the point or line, and pressing the tip against the tablet surface. This action causes the inner sensor shaft of the stylus to retract approximately 1/32 inch, making contact with an internal switch which transmits the point coordinates.

No flag buttons are available on the pen stylus. The stylus is designed principally for high speed data entry applications, or rough applications where high-speed manual operator movement and hardcopy are desired. The pen stylus generally is not as effective as the cursor for applications in which accuracy is important.

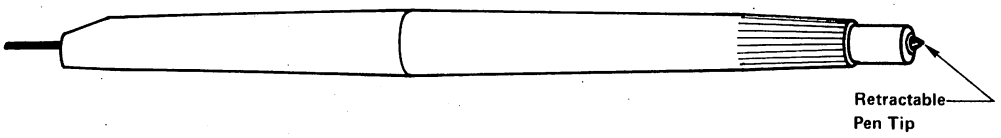

Figure 2-2. The Pen Stylus

The ballpoint pen refill in the stylus can be changed when the ink dries out or a new color is needed. Appendix C explains the procedure for replacing a stylus refill.

2.2 DIGITIZER CONTROL BUTTONS

The front panel of the digitizer chassis presents a sequence of pushbutton switches and a slider bar switch (see Figure 2-3). The significance and function of each control switch is explained below.

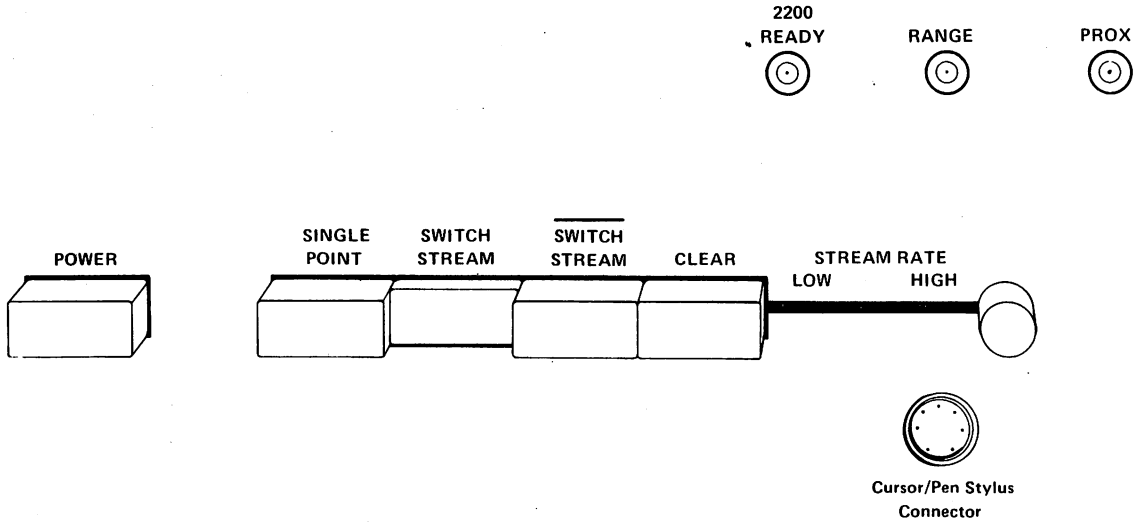

Figure 2-3. The Front Panel of the Digitizer chassis.

POWER

The POWER button controls power input to the digitizer.

In - Power ON.
Out - Power OFF.

SINGLE POINT, SWITCH STREAM, $\overline{\text{SWITCH}}$ STREAM

The SINGLE POINT, SWITCH STREAM, and $\overline{\text{SWITCH}}$ STREAM pushbuttons are used to select the desired mode of digitizer operation. The switches are ganged so that depressing one button automatically releases the others. The three modes of operation are explained in detail in Chapter 3, Section 3.4. Briefly, they operate in the following ways:

SINGLE POINT - Each depression of a cursor flag button or of the pen stylus, transmits one data point (one set of coordinates) to the system (provided system is ready to receive input from the digitizer).

SWITCH STREAM - A continuous stream of points (coordinate pairs) is sent to the system as long as a cursor flag button or the pen stylus remains depressed. The rate at which points are transmitted depends upon the position of the Stream Rate Slider Switch (see below), and the ready condition of the system (i.e., the processing time required by the controlling program for each point).

$\overline{\text{SWITCH}}$ STREAM - A continuous stream of points is sent to the system as long as the cursor or pen stylus is within 3/16 inch (0.5 cm) of the active surface of the digitizer tablet. The rate is determined by the position of the Stream Rate Slider Switch, and the ready condition of the system. When one of the cursor buttons or the pen stylus is depressed, the sign of the number representing the X and Y coordinates is plus (+); when the button or stylus is not depressed, the sign is minus (-).

CLEAR

CLEAR is a momentary pushbutton which clears and resets the digitizer's internal data registers.

STREAM RATE Slider Switch

The STREAM RATE Slider Switch determines the rate at which coordinate pairs are digitized and sent to the system in Switch Stream and $\overline{\text{Switch}}$ Stream modes. At the extremity of the LOW position, points are generated at a rate of approximately 5 points/second. At the extremity of the HIGH end, the rate is about 200 points/second. Positions between the two extremities provide intermediate rates.

Although the slider switch is used to control the rate at which points are digitized, the actual rate at which points are read into the system is determined by the amount of time required to process individual points as they are received. Processing time is primarily a function of the controlling program.

2.3 DIGITIZER CONTROL LIGHTS

In addition to the control switches, the chassis front panel contains three control lights, 2200 READY, RANGE, and PROX (refer to Figure 2-3).

2200 READY Light

The 2200 READY light is illuminated whenever the System 2200 is ready to accept a point from the digitizer (e.g., awaiting input from the digitizer). The READY light is extinguished when the System 2200 is processing any statement other than an input statement used to read data from the digitizer. While the 2200 READY light is out, no digitized points can be sent.

RANGE Light

The RANGE light is illuminated when the cursor or stylus is outside the legal digitizing range (i.e., raised more than 3/16 inch above the tablet surface, or positioned on the tablet outside of the active digitizing surface; the active digitizing surface of the tablet extends to within about 7/8 inch of the tablet margins on all sides). The RANGE light is extinguished when the cursor or stylus is within the valid digitizing area. When a range error is signalled, digitizing and transmission to the System 2200 are inhibited.

PROX Light

The PROX (proximity) light is, essentially, the converse of the RANGE light. It is illuminated in Switch Stream and Switch Stream modes as long as the cursor or pen stylus is positioned within the valid digitizing area (inside the active digitizing area, within 3/16 inch of the tablet surface). Illumination of the PROX light therefore indicates that the cursor or pen stylus is properly positioned to digitize and transmit points. Note that the PROX light is inoperative in Single Point mode.

CHAPTER 3

DIGITIZER OPERATION AND READOUT FORMAT

3.1 HOW THE MODEL 2262 "DIGITIZES"

In order to determine the coordinates of a point on a document placed upon the digitizer tablet, the digitizer must be able to establish the position of the cursor or stylus on the tablet surface with extreme accuracy. Although a number of techniques have been developed to pinpoint the location of a pen or cursor on the tablet, the method used by the Model 2262 is at once more elegant, efficient, and accurate than most others.

Embedded beneath the vinyl digitizing surface of the tablet are two arrays of small wires. One array of wires runs parallel to the X axis, while at right angles a second array runs parallel to the Y axis (see Figure 3-1). Together, the two arrays form a coordinate grid. Each array is electrically pulsed by a separate drive wire, which in turn is plugged into the digitizer chassis.

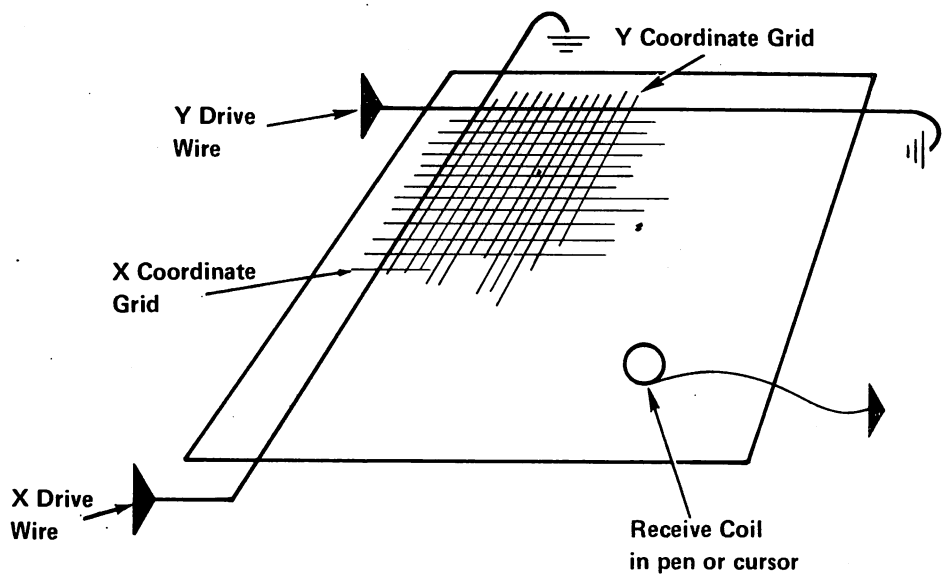

Figure 3-1
The Digitizer Tablet, Showing the Underlying Grid of X and Y Wires

When a cursor pushbutton or the pen stylus is depressed, the digitizer generates an electrical pulse from the X and Y drive wires along the X and Y wire arrays. As it moves along the wires, the electrical pulse induces a physical distortion of the ferromagnetic material of which the wires are constituted; this physical distortion is called a "strain wave," and it moves along the entire array of wires in a single front at a speed equal to the speed of sound for the wires. When the two strain waves moving along the X and Y arrays pass beneath the digitizing implement, they are detected by a receiving coil in the stylus or cursor.

By clocking and counting the time required for the strain waves to move from each drive wire to the receiving coil, the digitizer is able to determine the distances from the digitizing implement to the X and Y drive wires. These X and Y distances, measured in inches and accurate to less than .01 inch, represent the X and Y coordinates of the point digitized. They are transmitted, along with some additional control information, to the System 2200.

3.2 DIGITIZER READOUT FORMAT

For each point digitized, the digitizer generates a four-digit X coordinate, a four-digit Y coordinate, a flag digit, and a sign character. The elements are combined to form a single 11-character number called a "readout number." A single, unique readout number is generated for each point digitized. The format is illustrated in Figure 3-2 below:

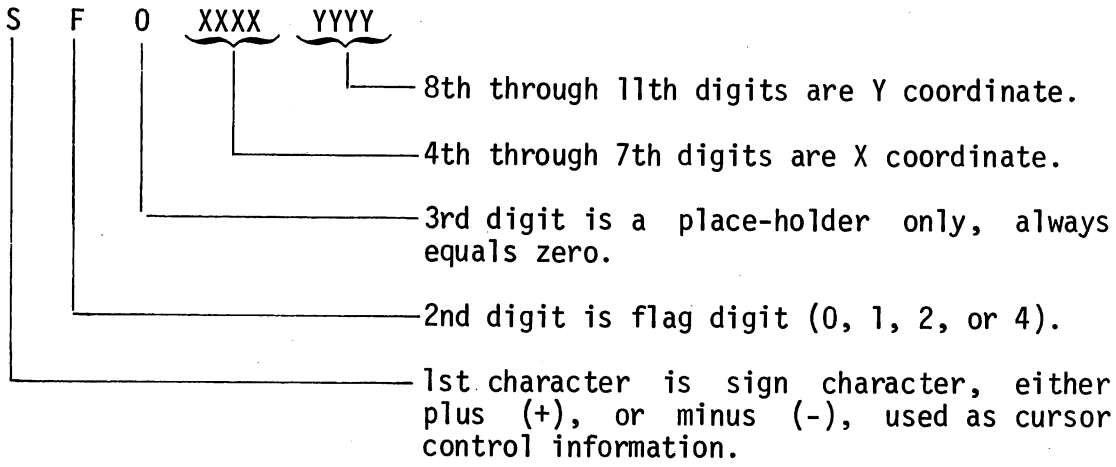

Figure 3-2. General Format of a Digitizer Readout Number.

A typical readout number looks like this:

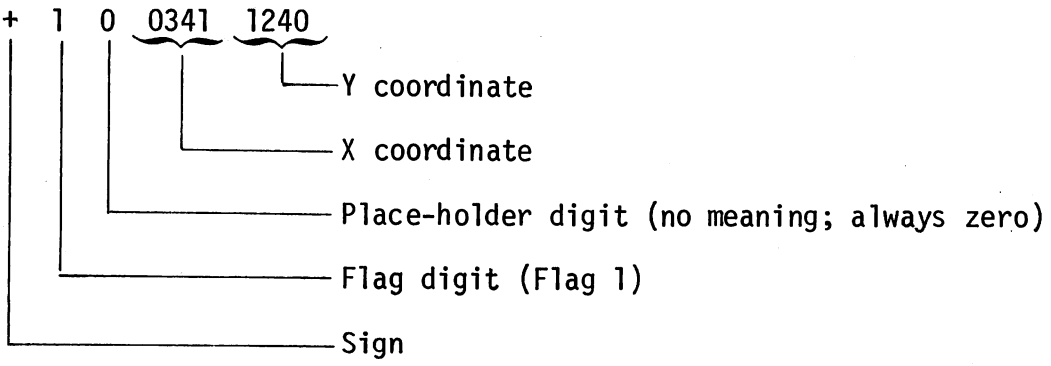

Figure 3-3. Typical Digitizer Readout Number

The readout value in Figure 3-3 was produced with the Flag 1 button on the cursor (since the flag digit has a value of 1). The X and Y coordinates represent the distances (in hundredths of an inch) of the cursor from the X and Y drive wires. The X coordinate in Figure 3-3, 0341,

signifies that the cursor in this case is 341 hundredths of an inch, or 3.41 inches, from the X drive wire. The Y coordinate similarly shows the cursor 1240 hundredths of an inch, or 12.4 inches, from the Y drive wire. The coordinates 0341 and 1240 are digitizer, or "absolute", coordinates, measured with respect to the underlying coordinate grid of the digitizer tablet. Before they can be meaningfully interpreted, the digitizer coordinates must be converted to "real" coordinates relative to the coordinate system of the user's document.

A more detailed discussion of the readout formats generated in Single Point, Switch Stream, and Switch Stream modes of digitizing is presented in the following section.

3.3 THE THREE MODES OF DIGITIZER OPERATION

The digitizer offers three modes of operation: Single Point, Switch Stream, and Switch Stream. The desired mode of operation is selected by depressing the appropriately marked pushbutton on the chassis front panel. Operating procedures and readout formats for each mode are described below in detail.

Single Point Mode

In Single Point mode, a single point is digitized and transmitted whenever a cursor flag button or the pen stylus is depressed (the system must be ready to accept a digitizer input at the time). If the system is not ready to receive when the point is transmitted (2200 READY light extinguished), the pushbutton or pen must be redepressed to transmit the point when the system is in a ready condition.

The sign of the readout number received by the system for each point in Single Point mode is always plus (+). In addition to the normal X and Y coordinate values, the first digit of the readout number is 0 if the Flag 0 pushbutton or pen stylus is depressed, or 1, 2, or 4 if the Flag 1, Flag 2, or Flag 4 pushbutton, respectively, is depressed.

If the cursor or pen stylus is positioned to generate a Range Error (outside the active digitizing surface, or more than 3/16 inch above the surface), the RANGE light illuminates, and no point can be digitized or transmitted. In systems equipped with the audio annunciator option, transmission of each point to the System 2200 causes the annunciator to emit an audible "beep".

Switch Stream Mode

In Switch Stream mode, a continuous series of points are digitized and transmitted to the system (if it is ready to receive) as long as a cursor flag button or the pen stylus is depressed. The digitized point stream is terminated when the button or stylus is released.

The rate at which points are digitized and transmitted in Switch Stream mode is continuously selectable from 5 points/second to 200 points/second with the Stream Rate slider bar. The effective rate at which

the controlling system accepts and processed points is, however, determined by the overhead time per point of the controlling program. The maximum stream rate of 200 points/second can be achieved if points are simply read directly into an array with DATALOAD, DATALOAD BT, or MAT INPUT, with no screening or intermediate processing. These statements and others are discussed in Chapter 4.

In Switch Stream mode, the readouts generated while the stylus or a pushbutton remains depressed conform to the format illustrated in Section 3.2. The flag digit of a series of readouts is determined by the button used in digitizing the corresponding points:

(1) Nothing depressed - No points transmitted.

(2) Flag 0 pushbutton - or pen stylus — Sign of readout number generally plus (+). First digit of number (flag digit) always zero. (Note: The sign of the last readout transmitted before the cursor pushbutton or pen stylus is released may sometimes be minus (-), due to transient electronic conditions.)

(3) Flag 1, 2, or 4 - Sign of readout number generally plus (+). The first digit (flag digit) of the readout number is 1, 2, or 4, depending upon whether Flag 1, Flag 2, or Flag 4 buttons, respectively, is depressed. (Note: The sign of the last readout transmitted before the Flag button is released may sometimes be minus (-), and/or the flag bit on the last readout may be omitted - i.e., the flag digit may be 0 - due to transient electronic conditions.)

When the cursor or stylus is positioned outside the legal digitizing range (i.e., outside the active digitizing surface, or more than 3/16 inch above the active surface), a Range Error is signalled, and no points can be digitized or transmitted.

The Switch Stream mode is useful for digitizing curves and other continuous graphical data in situations where only a selected portion of the curve or line must be digitized. Switch Stream mode might be used, for example, to stream in a burst of points which can be received at a high rate with the DATALOAD BT statement. If it is necessary to undertake extensive curve or line tracing, however, Switch Stream mode (described in the following paragraphs) may be more appropriate than Switch Stream.

Switch Stream Mode

Like Switch Stream mode, Switch Stream mode enables the digitizer to broadcast a continuous series of points to the receiving system. In Switch Stream mode, however, points are digitized and transmitted continuously as long as the cursor or stylus is within legal digitizing range (inside the tablet margins, within 3/16 inch of the tablet surface). It is not

necessary to depress the pen stylus or a cursor button in order to initiate digitizer operation in Switch Stream mode.

The rate at which points are digitized and transmitted to the receiving system is determined by the position of the Stream Rate slider bar. (The rate may vary from 5 points/second to 200 points/second.) However, points are actually received and processed by the system at a rate determined by the control program's overhead time per point. The maximum rate of 200 points/sec can be supported with the DATALOAD, DATALOAD BT, or MAT INPUT statements, provided the points are read directly into a receiving array, with no intermediate testing or processing.

Although points are generated in Switch Stream mode irrespective of whether the stylus or a pushbutton is depressed, the readout format can be affected by depressing the stylus or one of the buttons:

(1) Nothing depressed-	Sign of readout number always minus (-); first digit (flag digit) of readout number always zero.
(2) Flag 0 pushbutton or pen stylus depressed-	Sign of readout number always plus (+); first digit of readout number (flag digit) always zero.
(3) Flag 1, 2, or 4 pushbutton depressed-	Sign of readout number always plus (+); first digit (flag digit) of the readout number is 1, 2, or 4, depending upon whether Flag 1, Flag 2, or Flag 4, respectively, is depressed.

Note that the plus (+) sign or flag bits in the flag digit remain set as long as the stylus or pushbutton remains depressed. Typically, a number of readouts are received in this status because manual operation of the stylus or button is in most cases much slower than the stream rate. Often a flag is used to signal a special condition to the controlling system (such as end-of-job, first point, etc.). Each incoming readout is checked for the flag and, when the flag is detected, the system is instructed to carry out some special processing (in plotting, for example, a flag may instruct the system to raise or lower the pen). Since the flag bit generally is present in more than one readout number in Stream mode, however, a few simple BASIC program lines may be needed to bypass subsequent readouts with the flag set (that is, to wait until readouts are received without a flag before normal processing is resumed).

When the cursor or stylus is positioned outside the legal digitizing range (outside the tablet margins or more than 3/16 inch above the tablet surface), a Range Error is signalled, and digitizing is inhibited. The Range Error situation is common when the pen stylus is used, since the stylus is frequently lifted more than 3/16 inch above the tablet surface. During such periods, no points are digitized or transmitted. Once the pen point is within 3/16 inch of the tablet surface (it need not actually touch the surface), readouts are generated with minus signs.

Switch Stream mode is an extremely flexible mode of digitizing which is particularly useful for digitizing curves and other continuous graphic data where large numbers of points must be digitized and transmitted. A particular advantage in this mode is the fact that cursor or stylus action or inaction can be indicated by the sign change in the readout numbers. The System 2200 can be programmed to accept and process points with a plus sign, for example, and to branch to do other processing as long as the points received have a minus sign. (A program loop can test for inputs from both the digitizer and keyboard.) In this way, the controlling system is never hung up awaiting input from the digitizer. Some programming techniques which can be used to receive and process points in Stream mode are discussed in Chapters 4 and 5.

CHAPTER 4

THE DIGITIZER STATEMENTS

4.1 GENERAL PROGRAMMING CONSIDERATIONS

Basic Statements Used with the Digitizer

Five BASIC statements are available to receive input from the digitizer:

INPUT
KEYIN
DATALOAD
DATALOAD BT
MAT INPUT

Note that not all of these statements are available on all systems. The System 2200A, for example, supports only the INPUT statement for digitizer operations. (Although the DATALOAD statement is available for tape cassette operations on the 2200A, it lacks the logic necessary for digitizer operations.) Similarly, the System 2200S and WCS/10 support only INPUT unless modified with one of the available options 22-24. MAT INPUT is available only as part of an optional Matrix ROM for most systems.

The selection of a particular BASIC statement to receive digitizer input must be guided by the requirements of the individual application, since each statement has certain characteristic features and idiosyncrasies which may make it particularly suitable for some applications, and less suitable for others. All five of the available statements are discussed in detail in the sections which follow, and their general forms are included in Appendix A. The following brief summaries are intended only to highlight the principle advantages and drawbacks of each statement.

- INPUT permits variables or array elements (alpha or numeric), but not entire arrays, to be specified in its argument list. INPUT is serviceable for Single Point operations, but it is less practical in Stream operations, since it does not allow the specification of complete arrays as arguments. INPUT has two characteristics which may be considered drawbacks in some applications. Firstly, it is not possible to address the digitizer directly in an INPUT statement. Instead, INPUT operations must be SELECTed to the digitizer with a SELECT INPUT statement, and subsequently reselected to the keyboard when keyboard input is required. This may be a minor inconvenience in some programs. Secondly, INPUT automatically "echoes" each readout received on the CRT screen (or on any other currently selected Console Output device). This, too, may be an inconvenience if the screen contains information or instructions which the programmer does not wish disturbed. The echo may be suppressed by SELECTing Console Output back to the digitizer (see Section 4.2 on INPUT).

- KEYIN is not a data entry statement. It is used in applications which require input from both the digitizer and the keyboard to "scan" those devices and determine when one of them is ready to transmit data. When KEYIN detects a data transmission, it initiates a branch to a specified program line, where the data is read with a data entry statement such as INPUT, DATALOAD, etc. KEYIN and the concept of "scanning" are covered in Section 4.3.

- DATALOAD may be used to receive digitizer readouts into variables or arrays (alpha or numeric). Because it permits the use of entire arrays as arguments, DATALOAD can be used for Stream operations. An idiosyncrasy of the DATALOAD statement makes it somewhat undesirable for Single Point operations, however: DATALOAD always lags one point behind the digitizer. That is, DATALOAD does not read in the first point until the second point has been digitized; does not receive the second until the third has been digitized, etc. A further idiosyncrasy of DATALOAD, which may be significant for operations in Switch Stream mode, is that it strips the sign character off the readouts. In Switch Stream and Single Point mode, the loss of the sign is not significant, since the sign is always plus (+) in any case. In Switch Stream mode, however, the sign may be either plus (+) or minus (-), and it is frequently used as a flag to signal special conditions in this mode. For such an application, DATALOAD is not recommended.

- DATALOAD BT may be used to receive digitizer readouts into a single alphanumeric variable or array. Numeric variables and arrays are not permitted as arguments in a DATALOAD BT statement. DATALOAD BT is suitable for both Single Point and Stream operations. DATALOAD BT is one of only two statements (the other is MAT INPUT) which can support the maximum stream transmission rate of 200 points per second. This maximum rate is achievable only if the points are read directly into an array, without any intermediate processing. An idiosyncrasy of DATALOAD BT is that it reads 12 characters rather than the usual 11 for each readout. The twelfth character is an ASCII Carriage Return, interpreted as a control character and suppressed by all other statements, but read as data by DATALOAD BT. DATALOAD BT is a most efficient data entry statement; note, however, that it cannot be used to read points in numeric format.

- MAT INPUT may be used to receive points into one or more alpha or numeric arrays. Variables are not legal as arguments in a MAT INPUT statement. MAT INPUT is especially useful for large-scale stream operations, since it permits the specification of multiple arrays in its arguments, and it will support the maximum stream rate of 200 points/second.

Numeric vs. Alphanumeric Receiving Variables

Apart from the selection of an appropriate BASIC statement to receive digitizer input, the most critical programming consideration is whether to receive the readouts in numeric or alphanumeric format. If an alphanumeric variable or array is specified as the receiving argument, each readout is received as a string of 11 ASCII characters (a sign character and 10 digits). Since each ASCII character occupies one byte of memory, the receiving alpha variable or array element must be dimensioned to 11 bytes in length. For DATALOAD BT, which automatically tacks on a Carriage Return character at the end of each readout number, 12 bytes are needed. If, on the other hand, a numeric variable or array is used to receive the readouts, each readout is automatically converted to Wang internal numeric format as it is received, and occupies the standard eight bytes for a numeric variable.

The major programming advantages and disadvantages of alphanumeric and numeric variables in several common types of operation related to processing readouts are summarized below:

1. Storage

   Numeric variables are clearly more efficient in their use of memory for storing readouts. Each numeric variable or array element requires only eight bytes in memory, while an alpha variable or array element must be dimensioned to at least 11 bytes in length to receive a readout number. (If readouts are read with DATALOAD BT, the alpha variable must be 12 bytes in length.)

2. Packing

   Storage requirements are further reduced if the readouts are compressed with the PACK statement. Since the PACK statement operates by extracting the number from a numeric variable and packing it into an alphanumeric variable, the use of receiving numeric variables/arrays is obviously more efficient if the entire readout (including flag and both coordinates) is to be packed. If the X and Y coordinates are to be separated before packing, alpha variables/arrays may be preferred as receiving arguments.

3. Separating X,Y Coordinates

   The process of separating the X and Y coordinates from a readout is simpler and marginally faster if the readouts are received in alphanumeric format (i.e., stored in an alpha receiving variable/array) than if they are in numeric format (i.e., stored in a numeric receiving variable/array).

4. Testing for Flags (0, 1, 2, 4)

   Flag testing is simpler and marginally faster for readouts in alpha format than for those in numeric format. For processing in

Switch Stream mode, in which the sign may be either positive or negative, a flag-testing with numeric-format readouts is a complex and inefficient procedure.

5. Testing for Sign (Plus or Minus)

   As with flag testing, sign testing is simpler and marginally more efficient when performed on readouts in alphanumeric format. For systems which are not equipped with the MAT INPUT statement, it is not possible to test for signs in numeric-format readouts, since the only statement other than MAT INPUT capable of receiving readouts in a numeric array is DATALOAD, which strips off the sign character.

Summary

Choice of alpha or numeric receiving variables/arrays must be dictated by the particular requirements of the user's own application. In general, numeric receiving arguments are more efficient in their use of memory, and are recommended if points are to be received and stored with a minimum of intermediate processing. If such common processing operations as separating the X and Y coordinates and testing for flags and/or signs are to be carried out, however, alphanumeric receiving arguments may be preferred, since such tests tend to be more easily performed on literal strings than on numeric values. Programming examples for all of the operations listed above are discussed in Chapter 5.

4.2 INPUT

```
SELECT INPUT 65A

INPUT variable [,variable....]
```

INPUT can be used to receive one or more readouts from the digitizer into one or more numeric or alphanumeric variables or array elements in memory. INPUT is most commonly used to accept individual points from the digitizer in Single Point mode. Prior to reading points with INPUT, the digitizer must be selected for INPUT operations with a SELECT INPUT 65A statement (where 65A is the device address of the digitizer).

Example 4-1: Reading a Value from the Digitizer with INPUT

```
10 SELECT INPUT 65A  (select digitizer device address for input
.                     operations.)
.
.
100 INPUT A          (Read a value, convert to numeric.)
    or                    or
 90 DIM A$11         (Dimension A$ to 11 bytes, read a value as 11-
100 INPUT A$         character string.)
```

Notice in this example that the alphanumeric variable, A$, was dimensioned to 11 bytes, in order to store all 11 characters in the digitizer readout. For an explanation of the 11-character readout format, see Chapter 3, Section 3.3.

When the INPUT statement is used to receive digitizer readouts, the information is echoed back to the Console Output device (usually the CRT) as it is received. That is, each readout number is automatically displayed as received even if no PRINT statement is used in the program. If you do not want the CRT display disturbed during input, Console Output can be temporarily SELECTed to the digitizer itself (SELECT CO 65A). In this case, the echo is suppressed. Remember, though, to SELECT Console Output operations back to the CRT (SELECT CO 005) or other output device before attempting to print or display information.

A potential drawback of the INPUT statement is that, while the digitizer is selected for input operations, no input can be entered from the keyboard. In a program which combines digitizer input with operator input from the keyboard, therefore, the program must SELECT input operations back and forth between digitizer (SELECT INPUT 65A) and keyboard (SELECT INPUT 001). A similar situation arises with Console Output operations if the echo must be suppressed during input.

Although INPUT is most commonly used to read individual values in Single Point mode, it can also be used to receive a number of points in Switch Stream or Switch Stream mode. Because it is not possible to specify entire arrays as arguments in an INPUT statement, however, each value received must be read into a separate receiving variable or array element. For this reason, INPUT is slower and much less efficient in stream operations than DATALOAD or DATALOAD BT. Owners of systems which do not have a DATALOAD or DATALOAD BT capability may nevertheless find it convenient to use INPUT for limited Stream operations.

Example 4-2: Reading a Series of Values from the Digitizer with INPUT

```
10 SELECT INPUT 65A        (Select digitizer device address for input
.                           operations.)
.
.
100 INPUT A,B,C,A$,B$,C$   (Read a series of values from the digitizer.)
```

4.3 KEYIN

KEYIN alpha variable, line number, line number

KEYIN is available as a standard language feature on all systems except the System 2200A.

The KEYIN statement is used to "scan" one or more input devices (typically, in this context, the digitizer and/or the keyboard) to determine when one of them is ready to transmit data. When KEYIN detects a

data transfer, it reads the first character of transmitted data into a specified alpha variable, and initiates a program branch to a specified line number. In most instances, the specified line contains a data entry statement which proceeds to read data from the device transmitting. If no data transmission is detected by KEYIN, the program drops through to the next program line and continues normal processing.

KEYIN is most frequently programmed in a loop which begins with a KEYIN statement to check a device for data transmission, proceeds with other processing if the device is not ready, and loops back to the KEYIN statement upon the completion of the processing routine. The process of repeatedly looping back to check a device until it is found to be ready is called "scanning." In order to monitor a device in this way with KEYIN, the device must first be SELECTed for INPUT operations. Before KEYIN can be used to scan the digitizer, for example, the digitizer must be SELECTed for INPUT operations with a SELECT INPUT 65A statement.

In the KEYIN statement, three items of information must be specified:

1. A receiving alphanumeric variable or array element, into which the first character (sign character) of the digitizer readout is stored when the digitizer begins to transmit.

2. The line number to which the program will branch if KEYIN determines that the digitizer is ready to transmit. This line should contain an input statement which will read data from the digitizer.

3. A second line number, which has a purpose when KEYIN is used to scan for input from the keyboard, but which is meaningless when KEYIN is used with the digitizer. Two types of inputs may be received from the keyboard - characters, and Special Function codes (generated by one of the 16 Special Function Keys arrayed across the top of the keyboard). When the keyboard is scanned by KEYIN, receipt of a Special Function character causes the program to branch to the second line number instead of the first. Since the digitizer is incapable of generating a Special Function character, however, the second line number is included only to satisfy the format requirements of the general KEYIN statement, and is ignored by the system when INPUT operations have been selected to the digitizer.

Example 4-3: Scanning for Digitizer Readouts with KEYIN

```
10 SELECT INPUT 65A          (Select the digitizer device address for
                              INPUT operations.)

20 KEYIN A$,100,100          (Check to see if the digitizer is ready to
        .                     transmit. If yes, read 1st character (sign)
        .                     into A$, branch to line 100. If no, drop
        .                     through for other processing.)
(other processing)
        .
        .
        .
90 GOTO 20                   (Return to check digitizer again.)
```

```
100 INPUT B$ (or INPUT B) (Read remaining 10 characters of readout
                           (minus sign character, which is stored in
                           A$) into alpha or numeric variable.)
```

Notice in this example that line number 100 is specified twice. Any line number can be used for the second line number, since it is ignored by the system when KEYIN is used with the digitizer.

If you wish to retain the sign character when scanning with KEYIN, the STRing function can be used to store the remaining ten characters of the readout into the alphanumeric variable which already contains the sign character. For example, line 100 of the program in example 4-3 might be altered as follows:

```
100 INPUT STR(A$,2,10)
```

In this case, the remaining ten characters of the readout are stored in bytes 2-10 of A$ (which already contains the sign character in byte 1).

In most applications, KEYIN is used to scan the keyboard rather than the digitizer. A typical application involves the use of KEYIN to scan the keyboard for operator intervention during a stream operation. In this case, the program loops continuously to check the keyboard following each point read in stream mode. The operator can intervene to terminate digitizing simply by touching a key on the keyboard. Such intervention may be used to indicate termination of job, to allow the operator to switch from stream mode to Single Point mode (if the job calls for a combination of modes), or to signal a special condition to the program (such as pen up, pen down, change pen, identify special points, etc.). Such a capability is particularly valuable in operations with the pen stylus, which lacks the flag buttons typically used to signal any special conditions.

Example 4-4: Scanning the Keyboard During Stream Operations with KEYIN

```
10 DIM A$ 12
20 DATALOAD BT (N=12)/65A,A$
30 PRINT A$
40 KEYIN Q6$,80,120
.
.
.   (process point)
.
.
70 GOTO 20
80 PRINT "OPERATOR INTERVENTION"
.
.
.
120 STOP "END OF PROGRAM"
```

In this routine, points are read from the digitizer in Stream mode with DATALOAD BT (DATALOAD BT is discussed in Section 4.5). As each point is read, it is displayed. Following the read, KEYIN is used to scan the keyboard. If the operator touches one of the regular keyboard keys, the

program branches to line 80, where operator intervention is permitted. Alternatively, a Special Function key causes the program to branch to line 120 and terminate program execution. If nothing is entered from the keyboard, the program drops through for normal processing, and returns to line 20 to read the next point.

4.4 DATALOAD

DATALOAD [#n, /65A,] argument list

Where:

#n = Logical file number to which digitizer device address has been assigned. (n is an integer from 1 to 6).

/65A = The device address of the digitizer.

If neither #n or /65A is specified, the default address for TAPE-class operations (assigned in a SELECT TAPE statement) is used.

argument list = {variable / array designator}, . . .

array designator = array name followed by closed parentheses (e.g., A$(), N(), etc.).

DATALOAD is available as a standard language feature in all systems except the 2200A, 2200S, and WCS/10. DATALOAD may be added to the System 2200S or WCS/10 as part of one of the Options 22, 23, or 24. Note that a version of the DATALOAD statement is available on the System 2200A, and on standard versions of the 2200S and WCS/10, for tape cassette operations. It must be emphasized that the logic employed in the DATALOAD statement for tape cassette operations in these systems is not adequate to support digitizer operations, and the DATALOAD statement cannot for this reason be used to access the digitizer. The additional DATALOAD logic provided to the System 2200S and WCS/10 by the Options 22-24 does support digitizer operations.

The DATALOAD statement is used to read one or more readout values from the digitizer into one or more alphanumeric or numeric variables or arrays in memory. If the receiving variable or array is numeric, each readout is automatically converted to numeric format as it is received. For alphanumeric receiving variables and arrays, each variable and array element must be dimensioned to a length of at least 11 bytes, since each readout number is 11 characters in length (see Chapter 3, Section 3.2). Readout values are sequentially assigned to the receiving variables as they are received from the digitizer; if an entire array is specified, it is filled row by row.

Because it is possible to specify entire arrays as arguments, the DATALOAD statement may be used either for Single Point or stream operations. The maximum stream rate of 200 points per second can be supported with DATALOAD if points are read directly into an alphanumeric array, with no intermediate processing. Numeric arrays slow down the process somewhat, since they require that the readout values be converted into 2200 numeric format. Numeric arrays also offer better storage efficiency, however, with eight bytes per readout as opposed to 11 bytes per readout in alphanumeric arrays.

The DATALOAD statement has two prominent and interrelated idiosyncrasies:

1. The sign is stripped off every readout value but the first.

2. The incoming readouts are always one point behind the digitizer.

These anomalies in the operation of the DATALOAD statement arise from the fact that DATALOAD is normally used to read data values from punched tape via the Model 2203 Punched Tape Reader. On punched tape, each value is followed by two control characters, a Carriage Return character, and a Line Feed character. The Model 2252A Controller Board used with the digitizer generates only a Carriage Return character following each readout, however. DATALOAD therefore appropriates the next character following the Carriage Return - which is the sign character of the next readout - as its second control character. Because the control characters are discarded following data transmission rather than stored as data, the sign characters of the second and all subsequent readouts are lost. Further, the incoming readouts are always one point behind the digitizer, because the first point cannot be received until the second point has been digitized and its sign character appropriated; the second point then cannot be transmitted until the third is digitized, etc.

The absence of a sign character is, it should be noted, significant only for operations in Switch Stream mode. In Single Point and Switch Stream modes, the sign is always plus, and its loss therefore has no meaning at all for the controlling program. Similarly, the fact that the system lags one point behind the digitizer with DATALOAD may be significant for Single Point operations (certainly it must be noted and taken account of), but it is not likely to have much impact in Stream operations, where large quantities of points are being dealt with.

Example 4-5: Reading Individual Points with DATALOAD

```
10 SELECT #3 65A
20 DATALOAD #3,A
```

In this short example, the digitizer device address, 65A, is first assigned to logical file number #3. The DATALOAD statement at line 20 references #3 rather than the device address itself, and reads one point into numeric variable A. Note that the first point is not stored in A until a second point has been digitized. The sign of the first point received is intact, but the signs of the second and all subsequent points are stripped off.

Example 4-6: Reading a Stream of Points with DATALOAD

```
10 DIM N(100)
20 DATALOAD/65A, N()
```

In this example, the digitizer device address, 65A, is specified explicitly in the DATALOAD statement. One hundred points are read into numeric array N(). The array is filled row by row. The signs of all readouts except the first are stripped off.

4.5 DATALOAD BT

```
DATALOAD BT  (N=x)     #n,       alpha variable
                       /65A,     alpha array designator
```

Where:

(N=x)	=	The number of data characters to be read from the digitizer into the receiving alpha variable or array. For an alpha variable, a single readout consisting of 12 characters* is received; thus N=12. For an alpha array, 'N' should equal the total number of characters to be received (i.e., total number of readouts x 12 characters* per readout).
#n	=	A logical file number (#1 - #6) to which the digitizer device address, 65A, has been assigned in a SELECT statement.
/65A	=	The device address of the digitizer.
alpha variable	=	An alphanumeric variable dimensioned to 12 bytes in length.
alpha array designator	=	An alphanumeric array name followed by closed parentheses (e.g., A$(), F$(), etc.). Array elements must be dimensioned to a length of 12 bytes.

*Each readout number consists of 12 characters, rather than the standard 11, when read with DATALOAD BT. See below.

The DATALOAD BT statement is available as a standard language feature on all systems except the 2200A, 2200S, and WCS/10. DATALOAD BT may be added to the language set of the System 2200S and WCS/10 as part of one of the Options 22, 23, or 24.

DATALOAD BT is used to read points from the digitizer into a single alphanumeric variable or array. Numeric variables and arrays are not permitted, nor are multiple arguments. DATALOAD BT supports Single Point

and Stream mode operations, and will support the maximum stream rate of 200 points per second provided the data is read directly into an array with no intermediate processing.

DATALOAD BT is a special statement which is used with other types of input devices (such as cassette tape drives, punched tape readers, and card readers) to read unformatted data. Due to its specialized nature, DATALOAD BT, unlike all other digitizer statements, does not recognize the privileged status of control characters. The ASCII Carriage Return control character generated automatically by the 2252A controller board for each readout is thus interpreted by DATALOAD BT as another data character rather than a control character, and is transferred along with the 11 data characters into the receiving argument in memory. For this reason, the receiving variable or array element must be dimensioned to a length of 12 bytes rather than the usual 11 when points are read with DATALOAD BT.

As a further consequence of its non-recognition of control characters, DATALOAD BT does not recognize discrete digitizer readout values as such. Instead, the incoming readouts are viewed as portions of a single continuous character string of indeterminate length. For this reason, it is important that the receiving variable, or the elements of the receiving array, be dimensioned to exactly 12 bytes in length. If the receiving argument is shorter than 12 bytes, the remaining characters of the readout are held and tacked onto the beginning of the next readout. If the receiving argument is longer than 12 bytes, the first few characters of the next readout are appropriated to fill the excess bytes.

The 'N' parameter is helpful in specifying with exactness the total number of characters to be read by DATALOAD BT. For operations in which a single alphanumeric variable is used as the receiving argument, the 'N' parameter should specify that exactly 12 characters are to be read each time the DATALOAD BT statement is executed.

Example 4-7: Reading a Single Point with DATALOAD BT

```
10 SELECT TAPE 65A
20 DIM A$12
30 DATALOAD BT (N=12)A$
```

In this example, console tape operations are SELECTed to the digitizer address (65A) at line 10; there is therefore no need to include a device address or file number in the DATALOAD BT statement itself. The receiving alpha variable, A$, is dimensioned to a length of 12 bytes at line 20, and the 'N' parameter in the DATALOAD BT statement specifies that 12 data characters are to be read (N=12).

Note that the 'N' parameter really is redundant, since the statement would terminate when 12 characters have been read into a 12-byte argument even without this parameter. The 'N' parameter does, however, provide a measure of safety should the program accidentally neglect to dimension the receiving variable. The default length of an alpha variable is 16, rather than 12, bytes. Without the 'N' parameter, the system will read a 12-character readout and the first four characters of the next readout (for a total of 16) into an undimensioned receiving variable. If '(N=12)' is

specified, however, the transmission terminates at 12 characters even if there are remaining unfilled bytes in the receiving variable.

For alphanumeric arrays, the 'N' parameter specifies the total number of bytes to be received in the entire array. The total number of bytes is equal to the total number of readouts to be received multiplied times 12 bytes per readout. If 100 readouts are to be stored in an array, for example, the total number of bytes to be received is 1200 (100 x 12). In this case, too, it is imperitive that the array elements be dimensioned to 12 bytes in length in order that each element receive a complete readout.

Example 4-8: Reading Points in Stream Mode with DATALOAD BT

```
100 DIM A1$(10,10)12
110 DATALOAD BT (N=1200)/65A,A1$()
```

The array A1$() is dimensioned at line 100 to contain 100 elements (10x10), each 12 bytes in length. In the DATALOAD BT statement at line 110, the 'N' parameter specifies that 1200 characters (100 x 12) are to be received. Each element of A1$() receives a single 12-character readout. The array is filled row by row.

4.6 MAT INPUT

```
                    numeric array name [(d1[,d2])]
MAT INPUT           alpha array name [(d1[,d2]) [length]]
```

Where:

d = expression specifying a new dimension ($1 \leq d_1, d_2 \leq 255$; default: $d_1 = d_2 = 10$)

length = expression specifying maximum length of each alpha array element ($1 \leq$ length < 55) (default = 16)

The MAT INPUT statement is available as a standard language feature on some Wang systems, and may be obtained as part of an optional ROM for most others. MAT INPUT cannot be supported by a System 2200A.

MAT INPUT is used to receive digitizer readout values into one or more numeric or alphanumeric arrays. MAT INPUT is especially valuable in stream operations where moderate to large numbers of points must be received at relatively high speed. MAT INPUT can support the maximum stream rate of 200 points/second if the points are read directly into an array, with no intermediate processing.

Before MAT INPUT can be used to receive data from the digitizer, the digitizer must be selected for input operations with a SELECT INPUT 65A statement (where 65A is the device address of the digitizer). In the MAT INPUT statement itself, it is necessary to specify only the names of the receiving arrays. Closed parentheses are not required to designate entire arrays, because all arguments are assumed to be arrays (variables cannot be used as arguments in a MAT INPUT statement).

A standard 11-character readout is read by MAT INPUT. Alpha array elements should therefore be dimensioned to 11 bytes in length. One notable feature of MAT INPUT is its ability to dimension or redimension arrays with or without a previous DIM or COM statement. There are, however, a number of stringent restrictions on the use of MAT INPUT to redimension arrays; refer to the discussion of MAT INPUT in your Reference Manual for a full explanation.

In the general case, all arrays referenced in a MAT INPUT statement must be dimensioned in a prior DIM or COM statement. An array which originally was set up in a DIM or COM statement may be redimensioned within a MAT INPUT statement, but only to decrease its size. The initially established size of an array can never be increased in a MAT INPUT statement. Further, a one-dimensional array cannot be redimensioned to a two-dimensional array, and vice-versa. If an array which has not previously been dimensioned at all is referenced in a MAT INPUT statement, it is implicitly assigned the default dimensions of 10 x 10. Alpha array elements default to 16 bytes.

Example 4-9: Reading Points in Stream Mode with MAT INPUT (Arrays Previously Defined)

```
10 DIM A$(25,25)11
.
.
.
100 SELECT INPUT 65A
110 MAT INPUT A$
120 SELECT INPUT 001
```

An alpha array A$() is dimensioned at line 10 to contain 625 elements (25x25), each 11 bytes in length. At line 100, INPUT operations are SELECTed to the digitizer. The MAT INPUT statement at line 110 reads 625 points into array A$(). Note that the closed parentheses which ordinarily signify a complete array are omitted. At line 120, INPUT operations are returned to the keyboard.

An array may be redimensioned within a MAT INPUT statement if its size is not increased, and if it is not changed from a one- to a two-dimensional array, or vice-versa.

Example 4-10: Reading Points in Stream Mode with MAT INPUT (Arrays Redimensioned)

```
10 DIM B(20,20), N(10,10)
.
.
.
.
100 SELECT INPUT 65A
110 MAT INPUT B(10,10),N
120 SELECT INPUT 001
```

In this exmaple, the size of the numeric array B() is altered within the MAT INPUT statement. B() is redimensioned from a 20 x 20 matrix (its original size assigned at line 10) to a 10 x 10 matrix. Array N() is left with its initial dimensions of 10 x 10. The MAT INPUT statement reads a total of 200 points (100 points to each array), automatically converting each readout to numeric format (eight bytes) as it is received.

If an array specified in a MAT INPUT argument list has not previously been dimensioned, it automatically receives the default dimensions (10x10 matrix).

Example 4-11: Reading Points in Stream Mode with MAT INPUT (Arrays Implicitly Dimensioned)

```
100 SELECT INPUT 65A
110 MAT INPUT A,B
120 SELECT INPUT 001
```

Because the arrays A() and B() have not previously been dimensioned prior to their use as arguments in the MAT INPUT statement, they both are automatically dimensioned as 10 x 10 matrices. The MAT INPUT statement at line 110 therefore reads a total of 200 points (100 into each array). Note that this technique would be grossly wasteful of memory if alpha arrays were used, since alpha array elements default to 16 bytes, 5 more than the 11 bytes needed to store a readout.

Rules for Use of MAT INPUT

In summary, the following rules apply to the use of MAT INPUT:

1. Arrays cannot be redimensioned to contain a larger total number of bytes than their initially dimensioned size.

2. One-dimensional arrays (vectors) may not be redimensioned as two-dimensional arrays (matrices), and vice-versa.

3. Arrays not previously referenced or dimensioned are implicitly defined as 10 x 10 matrices when referenced in MAT INPUT. Alpha array elements default to 16 bytes in length.

4. Arrays dimensioned explicitly in a MAT INPUT statement must be matrices (two-dimensional arrays), and the total number of elements in each array may not exceed 100. The array(s) must be referenced for the first time in the MAT INPUT statement.

For a more exhaustive discussion of the Matrix ROM statements, and a complete list of the rules governing their use, refer to the Matrix ROM Reference Manual.

CHAPTER 5

FURTHER PROGRAMMING SUGGESTIONS

5.1 INTRODUCTION

This chapter presents and discusses a number of programming examples for some typical routines used in processing digitizer readouts. The topics covered include:

- Testing for Flags (Section 5.2)
- Testing for Plus/Minus Sign (Section 5.3)
- Separating X,Y Coordinates from Digitizer Readouts (Section 5.4)
- Testing for Increments of X,Y (Section 5.5)
- Packing Readout Numbers (Section 5.6)

5.2 TESTING FOR FLAGS

The three Flag buttons (Flag 1, Flag 2, Flag 4) on the four-button cursor are useful for signalling special conditions to the controlling system. Each button generates a unique digit (1, 2, or 4) as the first digit of the readout value. Incoming readouts can be tested for the presence of a flag digit; when one is detected, the program is instructed to branch to a routine which deals with the special condition, whatever it may be. (Initial point, last point, program termination, operator intervention, etc., are some conditions commonly signalled with flags.)

Techniques employed in testing for flags are of two basic sorts, depending upon whether the readout is in numeric or alphanumeric format. For readouts in alphanumeric format, the STRing function is used to test the second character (the first character is, of course, the sign). This technique is illustrated in Example 5-1. For readouts in numeric format, the process is somewhat more complex. A numeric readout is a discrete number of ten digits or fewer, whose leading digit is a flag digit if one of the flags 1, 2, or 4 is set, but not otherwise (since leading zeroes are dropped from a numeric value, a Flag 0 digit is not carried as part of the readout number).

A numeric readout is therefore tested for the presence of a flag digit by determining whether its value equals or exceeds a minimum value determined by the value of the first digit. For example, the minimum value generated by a Flag 1 button is 1000000000, or 1E9. In this case, the flag digit is one and both coordinates are zero. Thus, a readout which is equal to or greater than 1E9 must contain a flag digit other than (either 1, 2, or 4). Similarly, a readout which is equal to or greater than 2E9 must have either a 2 or 4 Flag digit; a readout equal to or greater than 4E9 must have a 4 flag digit.

Readouts may be tested to detect the presence of any flag digit (in which case all flags indicate the same condition), or to identify individual flag digits (in which case each flag has a unique meaning).

Example 5-1: Testing for Any Flag (Alpha Format)

```
100 DATALOAD BT/65A,B$
110 IF STR(B$,2,1) <> "0" THEN 200
.
.
.
```

At line 110, the second character of the readout (the first character is the sign) is tested for a value other than zero. If this digit is not zero, then it must be one of the flag digits 1, 2, or 4, and the program branches to a special routine starting at line 200.

Note that in checking for flags in numeric readouts, the number 1E8 serves as a convenient test number. The value 1E8 can never be generated as a readout number, because the second digit of a readout number is a "place-holder" which is always set to zero. The value 1E8 does, however, provide a clear line of demarcation between readouts with a zero flag and readouts with any other flag. A value smaller than 1E8 must have a zero flag digit, while a readout greater than 1E8 must have a 1, 2, or 4 flag digit.

Example 5-2: Testing for Any Flag (Numeric Format)

```
100 INPUT N
100 IF N > 1E8 THEN 200
```

The readout entered at line 100 is automatically converted to numeric format for storage in numeric variable N. At line 110, this number is compared with the test value 1E8. If the readout number is greater than 1E8, its first digit must be one of the flag digits 1, 2, or 4, and the program branches to a special routine. Otherwise, the flag digit is 0, and the program drops through for normal execution.

Example 5-3: Testing for Individual Flags 1, 2, or 4 (Numeric Format)

```
100 INPUT N
110 IF N>=4E9 THEN 200 (test for flag 4)
120 IF N>=2E9 THEN 250 (test for flag 2)
130 IF N>=1E9 THEN 300 (test for flag 1)
```

In this case, N is checked for individual flags, and the program branches to a different routine for each flag. For a Flag 0, the program drops through to continue processing at line 140.

Individual flags can be checked for readouts in alpha format with a series of STRing functions. A slightly more efficient routine, however, involves the use of the ON-GOTO statement.

Example 5-4: Testing for Individual Flags 1, 2, or 4 (Alpha Format)

```
100 DATALOAD BT/65A, A$
110 CONVERT STR(A$,2,1) TO A
120 ON A GOTO 200, 250, 130, 300
```

In this example, the flag digit is first converted to numeric format (line 110), then used in the ON-GOTO statement at line 120 to direct program flow. If A=1, the program branches to line 200; if A=2, a branch is made to line 250. Although there is no case where A can equal 3 (since there is no flag 3), the third line number specified is simply a place holder, and is never accessed; in this case, it is simply the next sequential line number (130). For a flag 4, the program branches to line 300. If the value of A is less than 1 (i.e., flag 0), program execution drops through to line 130.

When digitizing points in stream mode, it frequently happens that several readouts are transmitted with a flag digit, since manual operation of the flag button generally is slower than the stream rate. In this case, it is possible that the multiple flags will cause the program to branch to the special processing routine repeatedly rather than only once. To avoid this problem, the program should reject all subsequent points with a flag digit, and resume processing only when a zero flag is once again encountered.

Example 5-5: Test for Flag; If Flag Found, Reject Points Until Flag Button Released

```
100 SELECT INPUT 65A
110 INPUT A$
120 IF STR(A$,2,1) <> "0" THEN 200
.
.   (Normal processing)
.
.
190 GOTO 110
200 (Special processing)
.
.
.
.
250 INPUT B$
260 IF STR(B$,2,1) <> "0" THEN 250
270 GOTO 130
```

In this routine, the point entered at line 110 is first checked for a non-zero flag at line 120. If a flag is not found, the program drops through to resume normal processing beginning at 130. If a non-zero flag is found, a branch is made to line 200 for special processing. Following execution of the special routine, the next point is read and checked for a flag (lines 250, 260). If the flag digit remains set at its non-zero value, the program continues reading and checking points. When a point with a zero flag is read, the program returns to line 130 for normal processing.

5.3 TESTING FOR PLUS/MINUS SIGN

Operations in Switch Stream mode may call for testing the sign of each readout. (Note that sign testing is meaningless for points read in Single Point or Switch Stream mode, since in both cases the sign is always plus.)

It should be emphasized that the sign of a digitizer readout does not indicate the direction of movement along the axes. The sign is really nothing more than an additional flag available only in Switch Stream mode. It may be useful in that mode particularly when digitizing with the pen stylus, which is not equipped with flag buttons.

In Switch Stream mode, readouts are generated with plus (+) signs as long as the activator switch in the pen stylus (or one of the cursor pushbuttons) remains depressed. When the switch or button is released, the sign of each readout is minus (-), and remains so until the activator switch or pushbutton is once again depressed.

Example 5-6: Testing for Plus/Minus Sign

```
100 DATALOAD/65A,A
110 IF A < 0 THEN 150
```

or

```
100 DATALOAD/65A,A$
110 IF STR(A$,1,1) = "+" THEN 150
```

This example illustrates two methods of testing for a sign, depending upon the readout format. In the first case, the readout is read into a numeric variable, A, and tested for a minus sign. In the second case, the readout is stored in alpha variable A$, and a check is made for a "+" character in the first byte.

5.4 SEPARATING X,Y COORDINATES FROM DIGITIZER READOUTS

In order to operate with the X and Y coordinates for purposes of plotting or analysis, it is generally necessary to separate the individual coordinates from each readout. For readouts in numeric format, this process involves the application of two simple algorithms.

To obtain the X coordinate, the readout is multiplied times .0001, and the INT of the product is taken. This operation has the result of truncating the last four digits of the readout number (i.e., the Y coordinate), leaving only the X coordinate. Note, however, that this technique assumes a zero flag digit, and will yield an invalid result if a non-zero flag is present. It is therefore imperative that a flag test be performed prior to separating the X and Y coordinates.

Once the X coordinate is obtained, the Y Coordinate may be found by multiplying the X coordinate times 10000, and subtracting the product from the readout number. This operation has the effect of truncating the first

four digits of the readout (which, assuming a zero flag digit, will be the X coordinate), leaving only the Y coordinate.

Example 5-7: Separating X,Y Coordinates (Numeric Readout)

```
200 SELECT INPUT 65A
210 INPUT A
220 X = INT (A*.0001)
230 Y=A-X*10000
```

In this example, a readout number is first read into numeric variable A. The X and Y coordinates are then separated and stored in variables X and Y, respectively, utilizing the technique described above.

If the readout is stored in an alphanumeric variable, it is generally necessary not only to separate the X and Y coordinates, but also to CONVERT both coordinates into numeric format (since mathematical operations and plotting can be performed only with numbers).

Example 5-8: Separating X,Y Coordinates from Alphanumeric Readout, and Converting to Numeric Format

```
100 DATALOAD BT/65A,A$
110 CONVERT STR(A$,4,4) TO X
120 CONVERT STR (A$,8,4) TO Y
```

At line 110, characters 4 through 7 of A$ (the X coordinate) are converted to numeric format and stored in variable X. At line 120, characters 8 through 11 (the Y coordinate) are converted and stored in variable Y.

5.5 TESTING FOR INCREMENTS OF X,Y

In many stream operations, it is desirable to accept only selected points for storage or analysis. A typical routine to test for a minimal increment of the X and Y coordinates, and reject points which fall between the specified increments, is illustrated in Example 5-9 below.

Example 5-9: Typical Continuous Loop Testing for Increments of X,Y

```
 90 SELECT INPUT 65A
100 INPUT A
110 IF A<0 THEN 250:              REM CHECK FOR MINUS SIGN
120 IF A>1E8 THEN 300:            REM CHECK FOR FLAG
130 X1=INT(A*.0001):              REM SEPARATE X COORD
140 Y1=A-X1*10000:                REM SEPARATE Y COORD
150 IF ABS (X-X1)<10 THEN 100:    REM CHECK ΔX
160 IF ABS (Y-Y1)<10 THEN 100:    REM CHECK ΔY
170 X=X1:                         REM UPDATE LAST ACCEPTED X COORD
180 Y=Y1:                         REM UPDATE LAST ACCEPTED Y COORD
.
. . (process point)
.
240 GOTO 100:REM RETURN FOR NEXT POINT
```

This example illustrates a typical loop which scans incoming readouts and accepts points at a given interval of X and Y. Before any separating or increment testing is done, however, the readout is checked for a minus sign and flag digit at lines 110 and 120. The X and Y coordinates are then separated out at lines 130 and 140, and are stored in variables X1 and Y1, respectively. Variables X and Y contain the last accepted X and Y coordinate values. The differences between X and X1, and between Y and Y1, represent the X and Y intervals; these are checked for a minimal value of 10. (Since the absolute coordinates are in units of hundredths of an inch, 10 digitizer units equals 0.1 inch. Points less than 0.1 inch apart are thus rejected by this routine.) If the intervals for both coordinates do not equal the minimum increment of 10, the point is rejected, and the program returns to line 100 to read another point. If the interval is met or exceeded, the new coordinates are stored in X and Y to serve as the last accepted coordinates for testing the next point, and normal processing is followed.

5.6 PACKING READOUT NUMBERS

In stream operations (such as curve tracing) which involve the storage of large quantities of points, it frequently is worthwhile to store the readout numbers in packed decimal format. If the flag digit is stored, each readout number requires five bytes of storage in packed format; without the flag digit, only four bytes are needed. Both cases compare favorably with the eight bytes per readout required for readouts in standard numeric format, or the 11 bytes required for readouts in alphanumeric format.

The technique of packing incoming readouts provides a means of reducing total storage requirements without extensive intermediate processing of each point, and is recommended for applications in which points must be streamed in at relatively high speeds and stored off without processing. Typically, the procedure is to pack points into a large array as they are received, and, when the array is filled, write it out to tape or disk. Packing permits a maximum number of points to be read before the stream must be interrupted to write out the array. (Note that the System 2200 does not have a multiprocessing capability; therefore while data is being written out to tape or disk, incoming points from the digitizer are ignored by the system and lost. Only when the disk or tape write operation is completed does the system return to the reception of incoming readouts.)

If flags are used for any purpose in the stream operation, they generally must be preserved when the readouts are stored, since they will eventually be needed when the data is recalled from storage for processing. This is clearly the case, for example, if flags are used for plotter control (indicating pen up, pen down, etc.). The flags must be stored off along with the coordinates, so that they can be recalled when the points are replotted.

Each readout digit requires one-half byte of storage in packed format. A readout number which includes a flag digit consists of 10 digits, and requires five bytes of memory in packed format. A readout number without flag (i.e., with a zero flag) consists of only eight digits, and requires four bytes of storage.

Example 5-10: Typical Input Routine with Packing

```
 5 DIM P$(252)5,A(252)
10 DATASAVE DC OPEN F #1,240,"POINTS"
20 DATALOAD/65A,A()
30 PACK (##########)P$() FROM A()
40 DATASAVE DC#1,P$()
50 GOTO 20
```

This example illustrates a compact routine for reading points in stream mode, packing them, and storing them out on disk in a data file named "POINTS". Points are first read into an alphanumeric array, A() (line 20); when A() is filled, the contents of entire array are packed into alpha array P$() (line 30). Each '#' character in the PACK image designates one digit; thus, 10 digits are packed from each element of A() into a corresponding five-byte element in P$(). When the packing operation is finished, the contents of P$() are written out on disk, and the program loops back to read in more points. The size of the array (252 elements) is designed to maximize disk storage efficiency and access time, since 252 five-byte fields make up exactly six sectors.

5.7 SOME MORE SOPHISTICATED PROGRAMMING TECHNIQUES

The examples discussed in the preceding section have involved the use of straightforward relatively simple techniques for reading and processing digitizer points. For the more advanced programmer, Wang BASIC offers a variety of special statements which are more complex and sophisticated than those previously discussed, but which may enhance overall program performance and data reduction efficiency in many applications. These statements are not available as standard features on most Wang systems, but may be obtained as parts of one or more optional ROMs.

- $PACK - Useful in compressing a number of digitized points into an array for storage on disk or tape.
- $UNPACK - Can be used to convert a number of points from an alphanumeric array into separate X,Y numeric values with a single statement.
- MAT SEARCH- May be useful for rapidly scanning an alpha array filled with points for a special termination character (such as a minus sign).

- $GIO - An alternate means of entering points from the digitizer which permits greater flexibility in the specification of a termination condition than standard digitizer statements. $GIO has a provision for time-out and/or termination by a special character such as a minus sign.

Their complexity precludes any detailed discussion of these statements in the present manual. The interested reader is directed to the System 2200 Reference Manual, or the appropriate ROM reference manual, for further discussion.

APPENDIX A

GENERAL FORMS OF THE DIGITIZER STATEMENTS

This Appendix contains the General Form of each digitizer statement, with a brief explanation of its operation. The statements are treated in alphabetical order for ease of reference:

DATALOAD
DATALOAD BT
INPUT
KEYIN
MAT INPUT

The General Forms of the statements in this appendix focus specifically upon the statements' use in digitizer operations.

DATALOAD

General Form: DATALOAD [#n, /xxx,] argument list

where #n = Logical file number to which a device address has been assigned (n is integer from 1 to 6).

xxx = Device address of paper tape reader.

If neither of the above is specified, the default device address (the device address currently assigned to TAPE (see SELECT)) is used.

argument list = {variable / array designator} ,...

array designator = array name ( ) e.g., A(), A$().

Purpose:

The DATALOAD statement reads values from a specified input device and stored them in the receiving variables and/or arrays in the argument list. To use DATALOAD for digitizer operations, the digitizer device address is specified, DATALOAD defaults to the currently-selected console tape device. Normally, the console tape device is the CRT tape cassette drive, address 10A. However, the digitizer can be designated as console tape device with a SELECT TAPE 65A statement. In that case, all DATALOAD operations default to the digitizer.

Digitizer readouts are received and sequentially assigned to receiving arguments in the argument list. Arrays are filled row by row. When all receiving variables and/or arrays have been filled, the System 2200 disregards further transmissions from the digitizer. An idiosyncrasy of the DATALOAD statement is its suppression of the sign characters for all but the first readout received. In alphanumeric variables and arrays, the sign character is received as a blank (space); in numeric variables and arrays, it is received as a zero. In all cases, DATALOAD causes the receiving system to lag one point behind the digitizer.

Examples:

1)
```
100 DATALOAD 65A, A$
```

2)
```
150 DIM A$(10,10)11,B$(5,5)11
160 DATALOAD/65A, A$(), B$()
```

3)
```
100 DIM N(20,20), P(20,20)
110 SELECT TAPE 65A
120 DATALOAD N(), P()
130 SELECT TAPE 10A
```

DATALOAD BT

DATALOAD BT [(N=expression)] [#n, | /xxx,] {alpha variable | alpha array designator}

Where	N =	The number of characters to be read. For a single variable, N=12, since each readout is twelve characters in length. For an entire array, 'N' must be set equal to the total number of bytes to be received (a multiple of 12) and each element must be dimensioned to 12 bytes in length.
	#n =	Logical file number to which the digitizer device address has been assigned ('n' is an integer from 1 to 6).
	/xxx =	Device address of digitizer (65A).
		If neither #n nor /xxx is specified, the statement defaults to the device address of the currently selected console tape device. Normally, this is the CRT cassette drive (address 10A).
alpha array designator =		An alphanumeric array name followed by closed parentheses (e.g., A$(), N1$(), etc.)

Purpose:

The DATALOAD BT statement receives readouts from the digitizer, and stores them in the specified alphanumeric variable or array. DATALOAD BT is used for digitizer operations by specifying the digitizer device address (65A) in the statement, or by designating the digitizer as console tape device (with a SELECT TAPE 65A statement). In the latter case, DATALOAD BT operations default to the digitizer even if no device address is specified.

The DATALOAD BT argument list is restricted to a single alphanumeric variable or array. If an array is specified, it is filled row by row. Readouts are received and stored until the variable or array is filled; subsequently, further transmissions from the digitizer are ignored. An idiosyncrasy of the DATALOAD BT statement is that it interprets the ASCII Carriage Return control character generated automatically by the controller board for each readout as a data character, and stores it along with the 11 readout characters in the receiving variable or array. For this reason, the receiving variable or array elements must be dimensioned to 12 bytes in length rather than 11. The twelfth and last character is always the Carriage Return.

Examples:

1)
```
500 DIM A$(20,20)12
510 DATALOAD BT (N=4800)/65A,A$()
```

2)
```
100 DIM N1$12
110 DATALOAD BT (N=12)/65A,N1$
```

3)
```
250 DIM F$(100,10)12
260 SELECT #1 65A
270 DATALOAD BT (N=12000)#1, F$()
```

4)
```
10 SELECT TAPE 65A
20 DIM N$12
30 DATALOAD BT (N=12)N$
```

INPUT

General Form: INPUT ["character string",] variable [,variable...]

Purpose:

The INPUT statement permits the operator to transmit data from the digitizer to the receiving system during program execution. INPUT can be used for INPUT operations (with a SELECT INPUT 65A statement). After all readouts have been received, INPUT operations should normally be returned to the keyboard (SELECT INPUT 001).

INPUT is well suited for single point operations, but its usefulness in stream operations is severely restricted by the fact that entire arrays cannot be used as arguments in an INPUT statement. Multiple variables can be specified in the argument list, however. In that case, readouts are received and assigned to the receiving variables sequentially. Data received with INPUT is automatically "echoed" on the CRT screen. If the CRT display must remain undisturbed during the input operation, the echo can be suppressed by designating the digitizer itself for console output (SELECT CO 65A). Following the input operation, console output operations should be returned to the CRT display (SELECT CO 005).

Examples:

1)
```
10 SELECT INPUT 65
20 INPUT A$
30 SELECT INPUT 001
```

2)
```
100 SELECT INPUT 65A
110 INPUT A,B,C,D
120 SELECT INPUT 001
```

KEYIN

General Form: KEYIN alpha variable, line number, line number

Purpose:

The KEYIN statement checks for a character in the output buffer of the currently selected input device. If a character is ready for transmission, it is read into the receiving alpha variable, and program execution branches to a specified line. KEYIN can be used to check the digitizer for readout transmission if the digitizer is selected for input operations (with a SELECT INPUT 65A statement). Depending upon whether a readout is awaiting transmission to the system, KEYIN takes one of the following actions:

1. NO READOUT READY - Program execution continues at the next statement line.

2. READOUT READY - The first character of the readout (the sign character) is stored in the receiving alpha variable, and execution continues at the first line number specified.)

3. READY WITH SPECIAL FUNCTION CHARACTER - Applies only to input from the keyboard; not meaningful when KEYIN used to scan the digitizer. If a Special Function Key code is the first character in the output buffer of the keyboard, this character is read into the alpha variable, and program execution resumes at the second line number specified. For digitizer operations, the second line number is never used, and must be specified only to satisfy the general format requirements of the KEYIN statement.

The KEYIN statement provides a convenient way to scan the digitizer and other input devices (such as the keyboard) to accept input from the device which is currently transmitting.

Examples:

1)
```
10 SELECT INPUT 65A
20 KEYIN A$, 200, 200
```

2)
```
50 SELECT INPUT 65A
60 KEYIN N$(1), 200, 200
70 GO TO 60
```

3)
```
40 SELECT INPUT 65A
50 KEYIN STR(A$,1,1),100,100
```

MAT INPUT

General Form:

MAT INPUT { numeric array name [(d_1 [d_2])] / alpha array name [(d_1 [,d_2])[length]] } [,...]

Where: d = expression specifying a new dimension ($1 \leq d_1, d_2 \leq 255$; default : $d_1 = d_2 = 10$)

length = expression specifying maximum length of each alpha array element ($1 \leq$ length < 65) (default = 16)

Purpose:

The MAT INPUT statement enables the operator to enter values into one or more arrays under program control. MAT INPUT can be used to receive data from the digitizer if the digitizer is selected for INPUT operations (with a SELECT INPUT 65A statement).

When the system encounters a MAT INPUT statement during program execution, it displays a question mark (?) and awaits data from the digitizer. Arrays in the argument list are filled row by row with digitizer readouts, until all arrays have been filled. When all receiving arrays are filled with data, the system ignores any subsequent transmissions from the digitizer.

Arrays may be redimensioned in the MAT INPUT statement by specifying the new dimensions in parentheses following the array name. Similarly, the lengths of alphanumeric array elements may be specified following the array dimensions. For digitizer operations, alpha array elements should be dimensioned 11 bytes in length. In general, arrays must be dimensioned initially in a DIM or COM statement preceding the MAT INPUT statement. In this case, the redimensioend array may not exceed its initially dimensioned size (in total bytes), nor may a two-dimensional array be redimensioned as a one-dimensional array, or vice-versa. In certain cases, arrays may be dimensioned initially in the MAT INPUT statement itself. Such arrays must be matrices (two-dimensional), they must be referenced initially in the MAT INPUT statement, and they may not exceed 1600 total bytes in size. If the dimensions of a newly-dimensioned array are not specified, the default dimensions 10 x 10 are applied automatically; alpha array elements default to 16 bytes in length.

Examples

1)
```
10 SELECT INPUT 65A
20 DIM A$(25,25)11, B$(25,25)11
30 MAT INPUT A$,B$
```

2)
```
10 SELECT INPUT 65A
20 DIM A$(25,25)11, B$(25,25)11, N(15,15)
.
.
.
100 MAT INPUT A$, B$(20,20), N(10,15)
```

3)
```
10 SELECT INPUT 65A
20 MAT INPUT A$, F$(20,5)11, P(5,5)
```

4)
```
10 INPUT "ARRAY DIMENSIONS" X,Y,Z
20 SELECT INPUT 65A
30 MAT INPUT A$(X,Y)Z
40 SELECT INPUT 001
```

APPENDIX B

RESTORING PROPER MAGNETIC BIAS TO THE DIGITIZER TABLET

The digitizer tablet has a permanently biased magnetic substrate under the vinyl digitizing surface. If magnetic objects or large pieces of metal are accidentally placed on the tablet, or if the tablet is exposed to a magnetic field whose intensity is greater than about 40 gauss, the bias can be affected. Alternation of the bias will reduce the digitizer's accuracy. Should you encounter this problem, you can easily restore proper bias to the tablet simply by wiping it with the bias magnet provided. Noe, however, that the magnet may mark the tablet surface. To protect the surface, cover it with newspaper or a sheet of plastic film before wiping it with the magnet.

Hold the magnet as shown in Figure B-1 below (the arrows on the label must point toward the tablet surface). With the magnet against the surface of the tablet, wipe diagonally from the upper lefthand corner to the lower righthand corner, being sure to cover the entire digitizing surface. When you have restored proper bias to the tablet, store the bias magnet away from the tablet, and away from all tape cassettes and disk platters.

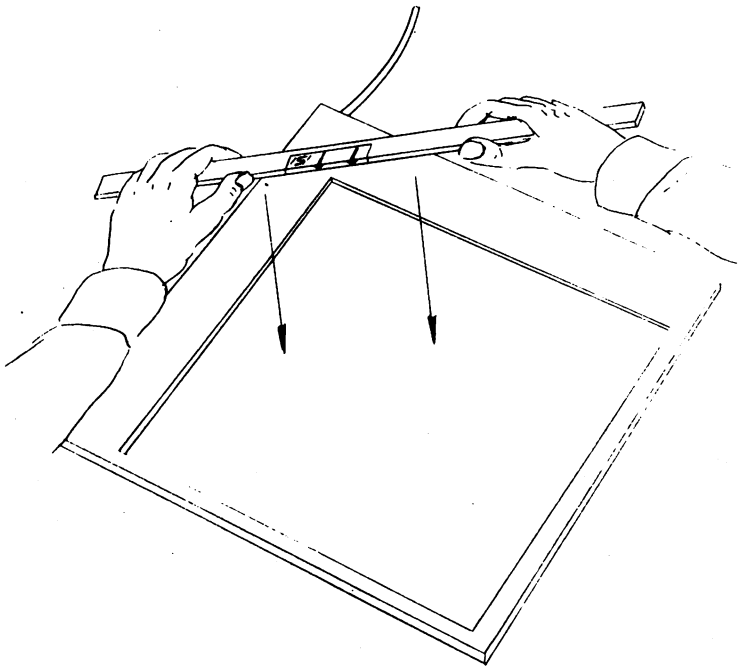

Figure B-1. Tablet Wiping Procedure

APPENDIX C

REFILLING THE PEN STYLUS

The ball point pen refill in the digitizer pen stylus can be replaced when it dries out or a change of color is wanted. To change or replace the refill, follow this procedure:

1. Unscrew the front section of the stylus.

2. Grasp the front internal section and carefully pull it straight out. (Note: The internal front sleeve may come off with the outer cover. If this happens, remove the inner sleeve before attempting to reassemble the stylus or you may damage the miniature connector.)

3. Pull out the pen refill and replace with a new one.

4. Replace the front inner section and mate connectors.

5. Replace the front outer cover.

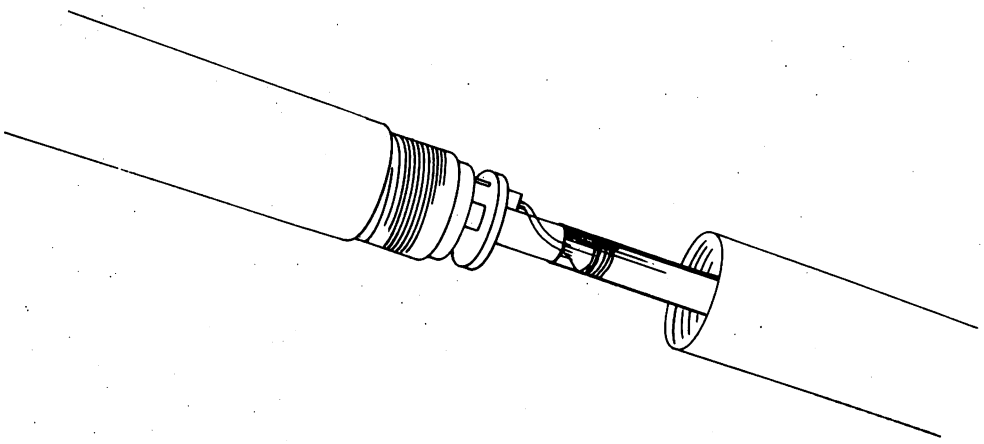

Figure C-1. Disassembled Pen Stylus

APPENDIX D

DIGITIZER SPECIFICATIONS

TECHNICAL DATA

Resolution
  100 lines/inch. Number of bits depends on tablet size.
Linearity
  .08% of full scale.
Accuracy
  Less than .01 inch.
Repeatability
  Less than .01 inch.
Stability
  Less than .01 inch.
Repetition Rate
  Continuously variable up to 200 coordinate pairs per second.*
Coding
  BCD
Coordinate Origin
  Lower-left corner (first quadrant operations).
Coordinates
  Absolute
Operating Modes
  Point - single coordinate pair transmitted by depressing pen or cursor button.
  Switch Stream - coordinate pairs transmitted continuously while pen or cursor within legal range.
  Switch Stream - coordinate pairs updated continuously while pen or a cursor button depressed.
Controls
  Pushbutton selection of Point, Switch Stream, or Switch Stream. Slider bar selection of stream rate. Three special flag buttons on the cursor for special control functions.
Indicator Lights
  RANGE - lights when cursor or pen outside active digitizing area.
  2200 READY - lights when the system is ready to receive points.
  (PROX(Proximity) - lights when stylus or cursor within 1/4" of the active tablet surface (i.e., within legal digitizing range).

Digitizing Elements
  Standard 4-button cursor with black bull's eye sight. Standard pen stylus.
Power Requirements
  0.3A, 115 VAC, 60 Hz
  0.15A, 220 VAC, 50 Hz
  50 Watts
Operating Environment
  Background Magnetic Field - 20 gauss max.
Temperature - 50°F to 90°F (10°C to 30°C)
Humidity - 20% to 80%

PHYSICAL SPECIFICATIONS

Digitizer Tablet Sizes
  2262-1 . . . . 20 in. x 20 in.
  2262-2 . . . . 30 in. x 40 in.
Digitizer Tablet (Weights)
  2262-1 . . . . 20 lb (9 kg)
  2262-2 . . . . 60 lb (27 kg)
  2262-3 . . . . 81 lb (37 kg)
Digitizer Chassis
  Height . . . 7.25 in. (18.12 cm)
  Width . . . 9.75 in. (24.38 cm)
  Depth . . . 11.75 in. (29.38 cm)
  Weight . . . 14 lb (6.3 kg)

* Maximum stream rate is a function of controlling program in the System 2200.

INDEX

To help us to provide you with the best manuals possible, please make your comments and suggestions concerning this publication on the form below. Then detach, fold, tape closed and mail to us. All comments and suggestions become the property of Wang Laboratories, Inc. For a reply, be sure to include your name and address. Your cooperation is appreciated.

700-3735

TITLE OF MANUAL: **2262 X-Y DIGITIZER REFERENCE MANUAL**

COMMENTS:

Fold

Fold

(Please tape. Postal regulations prohibit the use of staples.)

WANG

Fold

FIRST CLASS
PERMIT NO. 16
Tewksbury, Mass.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

– POSTAGE WILL BE PAID BY –

WANG LABORATORIES, INC.
836 NORTH STREET
TEWKSBURY, MASSACHUSETTS 01876

Attention: Marketing Department

Fold

Cut along dotted line.

Printed in U.S.A.

Walter Jackson

WANG LABORATORIES (CANADA) LTD.
49 Valleybrook Drive
Don Mills, Ontario M3B 2S6
TELEPHONE (416) 449-2175
Telex: 069-66546

WANG EUROPE, S.A.
Buurtweg 13
9412 Ottergem
Belgium
TELEPHONE 053/704514
Telex: 26077

WANG DO BRASIL COMPUTADORES LTDA.
Rua Barao de Lucena No. 32
Botafogo ZC-01 20,000
Rio de Janeiro RJ
Brasil
TELEPHONE 226-4326
Telex: 2123296

WANG COMPUTERS (SO. AFRICA) PTY. LTD.
Corner of Allen Rd. & Garden St.
Bordeaux, Transvaal
Republic of South Africa
TELEPHONE (011) 48-6123

WANG INTERNATIONAL TRADE, INC.
836 North Street
Tewksbury, Massachusetts 01876
TELEPHONE (617) 851-4111
TWX 710-343-6769
Telex: 94-7421

WANG SKANDINAVISKA AB
Pyramidvaegen 9A
S-171 36 Solna,
Sweden
TELEPHONE 08-826814
Telex: 11498

WANG NEDERLAND B.V.
Damstraat 2
Utrecht, Netherlands
(030) 93-09-47
Telex: 47579

WANG PACIFIC LTD.
902-3 Wong House
26-30, Des Voeux Road, West
Hong Kong
TELEPHONE 5-435229
Telex: 74879

WANG INDUSTRIAL CO., LTD.
110-118 Kuang-Fu N. Road
Taipei, Taiwan
TELEPHONE 784181-3
Telex: 21713

WANG GESELLSCHAFT M.B.H.
Formanekgasse 12-14
A-1190 Vienna, Austria
TELEPHONE 36.32.80
Telex: 74640

WANG S.A./A.G.
Markusstrasse 20
CH-8042 Zurich 6
Switzerland
TELEPHONE 41.1.26.6866
Telex: 59151

WANG COMPUTER PTY. LTD.
55 Herbert Street
St. Leonards, 2065
Australia
TELEPHONE 439-3511
Telex: 25469

WANG ELECTRONICS LTD.
1 Olympic Way, 4th Floor
Wembley Park,
Middlesex, England
TELEPHONE 01/903/6755
Telex: 923498

WANG FRANCE S.A.R.L.
Tour Gallieni, 1
78/80 Ave. Gallieni
93170 Bagnolet, France
TELEPHONE 33.1.3602211
Telex: 68958F

WANG LABORATORIES GMBH
Moselstrasse 4
6000 Frankfurt AM Main
West Germany
TELEPHONE (0611) 252061
Telex: 04-16246

WANG COMPUTER SERVICES
836 North Street
Tewksbury, Massachusetts 01876
TELEPHONE (617) 851-4111
TWX 710-343-6769
Telex: 94-7421

24 Mill Street
Arlington, Massachusetts 02174
TELEPHONE (617) 648-8550

WANG LABORATORIES, INC.

836 NORTH STREET, TEWKSBURY, MASSACHUSETTS 01876, TEL. (617) 851-4111, TWX 710 343-6769, TELEX 94-7421

Printed in U.S.A.
700-3735
10-75-1M