WANG

MODEL 2252A SCANNING INPUT INTERFACE CONTROLLER USER MANUAL

System 2200

2252A

Scanning Input Interface Controller

(BCD 1 to 10 DIGIT PARALLEL)

User Manual

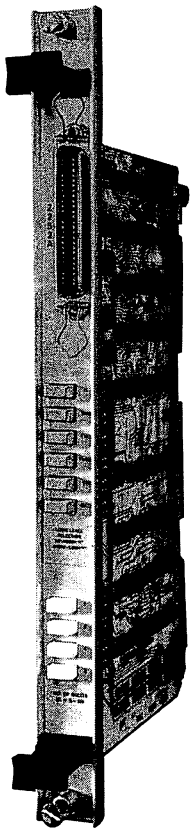
WANG LABORATORIES, INC.
ONE INDUSTRIAL AVENUE, LOWELL, MASSACHUSETTS 01851, TEL. (617) 851-4111, TWX 710 343-6769, TELEX 94-7421

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., assumes no responsibility for wiring the male connector, furnished with this interface controller board, to any non-Wang equipment. In no event shall Wang Laboratories, Inc., or its subsidiaries be liable for any damages in connection with or arising from the wiring or operation of non-Wang equipment interfaced to a Wang system.

WANG LABORATORIES, INC.
ONE INDUSTRIAL AVENUE, LOWELL, MASSACHUSETTS 01851, TEL. (617) 851-4111, TWX 710 343-6769, TELEX 94-7421



PREFACE

Information regarding the installation and operation of the Model 2252A Scanning Input Interface Controller are provided in this manual.

Chapter 1 discusses the special features of the Model 2252A controller and describes how digital meters and instruments are interfaced to a Wang System 2200.

Chapter 2 discusses programming techniques for operation of the Model 2252A controller.

Readers of this manual, particularly Chapter 2, should be familiar with the System 2200 and its BASIC language. Anyone not already familiar with the System 2200 should read the BASIC Programming Manual provided with each system before proceeding with Chapter 2, and should use the Reference Manual as a supplement to this manual.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EXAMPLES

CHAPTER 1

MODEL 2252A FEATURES

1.0 GENERAL DESCRIPTION

The Wang Model 2252A Scanning Input Interface Controller (BCD 1-To-10-Digit-Parallel) is a controller board which plugs into one of the I/O slots in any System 2200 Central Processing Unit (CPU) chassis. A 50-pin female Amphenol connector on the faceplate of the controller board facilitates direct connection of a device or instrument to the System 2200.

A 50-pin male Amphenol connector is supplied with the controller board. Responsibility for wiring the male connector to the cable from a non-Wang device is not assumed by Wang Laboratories.

The Model 2252A input-only interface is directly compatible with most digital meters for on-line applications. The interface can accept readouts with a sign (+ or -) and up to ten decimal digits (0 through 9) in standard BCD (Binary Coded Decimal) 8-4-2-1 notation; that is, four bits per decimal digit. Alternatively, the interface can accept up to forty-one discrete binary bits.

Four push button switches on the faceplate of the Model 2252A controller board can be set to indicate the exact number of BCD digits, any number from 1 to 10 digits (or the number of 4-bit groups of discrete binary data) to be strobed into the CPU for each readout. The sign bit is strobed into the CPU regardless of the setting on the "number-of-digits" push buttons.

The sign and BCD digits (or the discrete binary data) in each readout are accepted in parallel at typical TTL/DTL* voltage levels. Acceptable levels for low signals from a device are between 0 and + 0.4 volts d.c.; acceptable levels for high signals from a device are between + 2.4 and 5.0 volts d.c.

Digital instrument designers do not use a standard convention when letting a pair of voltage levels ("low" and "high") represent a two-state condition such as "true" or "false", or logic "0" or logic "1". Some designers let a low-signal denote logic "0" and a high-signal denote logic "1"; other designers reverse the definition. However, six push button switches (IS, SIGN, DATA, EXEC, TRANS, and EOT) on the faceplate of the Model 2252A controller board provide a logic-level-selection capability which makes a wide range of digital instruments compatible with the System 2200. Logic level selection is available for the following signals:

1. input strobes or level transitions from the interfaced device, indicating availability of a readout,

2. the sign bit in a readout,

---

*TTL = transistor-transistor-logic, DTL = diode-transistor-logic

3. the data bits in a readout,

4. "execute" signals from the interface, requesting a readout,

5. "transfer-in-progress" (busy) signals from the interface, indicating the current readout should not be changed,

6. "end-of-transfer" output strobes from the interface, indicating termination of data transfer to the CPU for the current readout.

During data transfer, a code converter in the interface converts the sign bit in a readout into an equivalent eight-bit ASCII code, where a plus is converted to a HEX(2B) code and a minus sign to a HEX(2D) code. Also, the code converter converts each four-bit code corresponding to one BCD digit into an equivalent eight-bit (two hexadecimal digit) ASCII code used by the System 2200. In the System 2200 BASIC language, the decimal digits 0 through 9 are represented by the codes HEX(30) through HEX(39), respectively. Therefore, a four-bit code representing one BCD digit determines the low-order hexadecimal digit of its equivalent ASCII code; the interface automatically supplies the high-order hexadecimal digit - - a hex three.

A parallel-to-series converter in the interface transfers the sign and decimal digits in a readout into the CPU, one-byte-at-a-time, in sequential order. First the sign byte is transferred. Then beginning with the most significant digit, each successive digit in a readout is transferred to the CPU until the total number of transferred digits equals the setting on the number-of-digits push buttons. Finally, the interface transfers a carriage-return character; that is, a HEX(0D) byte, denoting termination of data transfer for the current readout.

To keep the design of the Model 2252A interface as simple as possible, information from an instrument is read into the CPU as an integer. However, both fixed and floating point readouts from interfaced instruments can be processed using programming techniques presented in Chapter 2.

Several BASIC language statements can be used to program input operations via a Model 2252A interface:

1. INPUT, available in all standard CPU's including the System 2200 A,B,C,S and T.

2. DATALOAD BT (device type=6), available in the standard System 2200 B, C and T, and in the System 2200S with Option 22, 23 or 24.

3. MAT INPUT, available in the standard System 2200T, in the System 2200 B and C with Option 1, and in the System 2200S with Option 21, 22, 23 of 24.

4. $GIO, available in the standard System 2200T, in the System 2200 B and C with Option 2, and the System 2200S with Option 23 or 24.

Unless the performance characteristics of a particular interfaced device restrict data transfer rates to the System 2200, up to 100 readings per second can be received using INPUT and MAT INPUT statements; up to 800 readings per second can be received using DATALOAD BT statements; up to 1000 readings per second can be received using $GIO statements.

Other BASIC language statements can be used to scan an on-line device interfaced via a Model 2252A controller. Statements for scanning operations are:

1. KEYIN, available in the standard System 2200 B,C,S and T.

2. $IF ON, available in the standard System 2200T, in the System 2200 B and C with Option 2, and in the System 2200S with Option 23 or 24.

During a scanning operation, the Model 2252A interface is enabled and, immediately thereafter, produces an "execute" signal level requesting a readout. The level remains in effect until an input strobe or level transition from the interfaced device indicates a readout is available. Meanwhile, if a readout is not available, the interface is disabled and program execution advances to the next statement. Therefore, a program can be designed to test periodically for receipt of a readout while a scanning mode of operation is in effect.

Still other BASIC language statements can be used to process data received via a Model 2252A interface. Several very powerful bit/byte manipulation statements (not available in the System 2200A) are available in the standard System 2200 B,C and T, and in the System 2200S with Option 22, 23 or 24. The statements simplify programming requirements for on-line applications involving the processing of discrete binary data. Also, the data conversion statements $TRAN, $PACK, and $UNPACK simplify programming requirements when processing discrete binary data; these statements are available in the standard System 2200T, in the System 2200 B and C with Option 2, and in the System 2200S with Option 23 or 24.

For many on-line applications, a System 2200A with a Model 2252A interface is adequate for the collection and subsequent processing of BCD data; however, a System 2200A is not recommended for applications requiring the processing of discrete binary data. Neither Option 1, the Matrix ROM (Read Only Memory) nor Option 2, the General I/O ROM are available for the System 2200A. Also, the KEYIN statement is not included in the System 2200A.

1.1 CONNECTOR PIN ASSIGNMENTS

The information included in this section is essential for anyone planning to wire the Model 2252A male connector to the cable from a non-Wang device or instrument. Other readers may wish to omit this section.

Functional assignments for the 50 pins in the Model 2252A Amphenol connector are listed in Table 1-1. An open pin connection for any input circuit is equivalent to a high-level signal for that circuit.

Table 1-1. Model 2252A Connector Pin Assignments*

Pin Number	Function	8-4-2-1 Position	Pin Number	Function	8-4-2-1 Position
01	D_0 (most significant digit)	1	37	D_7 (eighth significant digit)	1
02		2	38		2
03		4	39		4
04		8	40		8
19	D_1 (second significant digit)	1	41	D_8 (ninth significant digit)	1
20		2	42		2
21		4	43		4
22		8	44		8
05	D_2 (third significant digit)	1	45	D_9 (tenth significant digit)	1
06		2	46		2
07		4	47		4
08		8	48		8
23	D_3 (fourth significant digit)	1	17	Sign bit	
24		2			
25		4	18	Input strobe	
26		8	31	Execute signal	
09	D_4 (fifth significant digit)	1	49	Transfer-in-progress	
10		2			
11		4	50	EOT output strobe	
12		8			
27	D_5 (sixth significant digit)	1	32	Prime strobe	
28		2	33	± 0 volts	
29		4	34		
30		8			
13	D_6 (seventh significant digit)	1	35	+ 5 volts	
14		2			
15		4	36	Chassis ground	
16		8			

*All logic is BCD 8-4-2-1 TTL compatible, positive true. Open input circuit pins are at high level. Pins 01 through 16, Pin 17, Pins 19 through 30, and Pins 37 through 48 can be used for input of up to 41 discrete data bits rather than BCD data.

The layout of the 50-pin Model 2252A connector for the System 2200 is compatible with the 36-pin connectors for the Model 705 and 605 microinterfaces used with the Wang 700 Series and 600 Series calculators. The functions assigned to Pins 01 through 36 in the Model 2252A connector are identical to the functions assigned to pins with the same numbers in either the Model 705 or the Model 605 connector. Therefore, if desired, an instrument formerly interfaced to a 700 or 600 Series calculator can be interfaced to the System 2200 by removing the 36-pin male connector and wiring the 50-pin Model 2252A connector to the instrument cable.

NOTE:

1. The Model 2252A interface can function properly if Pins 37 through 48 for input of three BCD digits (not included in the 36-pin Model 605 and 705 connectors) are tied to the ± 0 volt circuits on Pins 32, 33, and 34, or the number-of-digits switches are set for input of seven or less BCD digits (see Section 1.4).

2. Pins 49 and 50 in the Model 2252A connector can be left unconnected if the control signals available on these pins (Transfer-in-progress and EOT) cannot be utilized by a particular device being interfaced to the System 2200.

1.2 SIGNAL DESCRIPTIONS

Like Section 1.1, the information in this section is included for readers who plan to wire the Model 2252A male connector to the cable from a non-Wang device or instrument. Other readers may wish to omit this section.

Pin 17 — Sign Bit - - One circuit is provided for the sign bit in each readout (or input of one discrete binary bit). A logic "1" signal level on Pin 17 is interpreted as a plus sign; logic "0" is interpreted as a minus sign. When the sign bit is transferred to the CPU, the code converter on the Model 2252A interface converts a logic "1" signal level into the eight-bit ASCII code HEX(2B) for a plus sign. The code converter converts a logic "0" signal level on Pin 17 into the ASCII code HEX(2D) for a minus sign. The Up/Down position of the SIGN switch on the controller faceplate determines the signal level definition for the Pin 17 circuit (see Section 1.3).

Pins 01-16, 19-30, 37-48 — Input Data - - Forty parallel circuits are provided for up to ten BCD digits (each represented by the 8-4-2-1 bit-positions specified in Table 1-1), or up to forty discrete binary bits per readout. When a readout is transferred to the CPU, the data bits are processed 4-bits at a time, sequentially, beginning with the bits representing the most significant BCD digit (see Table 1-1). The code converter on the interface converts each four-bit code into an eight-bit (two hexadecimal digit) ASCII code by supplying a hexadecimal three $(0011)_2$ code as the high-order hexdigit and letting the incoming 4-bit code determine the low-order hexdigit. Conversion and transfer of the next sequential 4-bit group in a readout ceases when the number of processed groups equals the setting on the number-of-digits push buttons on the controller faceplate (see Section 1.4). The Up/Down position of the DATA switch on the controller faceplate determines the signal level definition for all forty input data pins (see Section 1.3).

Pin 31

Execute Signal - - The outgoing "initiation" or "request" level on Pin 31 is set to logic "1", when the Model 2252A interface (and the System 2200) is ready to receive an input strobe (or level transition) from the interfaced device. The level is reset to logic "0" five microseconds after an input strobe (or level transition) is received from the interfaced device. In general, for many interfaced instruments, a logic "1" level on Pin 31 requests a data readout or initiates a settling condition in the interfaced instrument. However, if an instrument is not designed to generate input strobes, Pin 31 can be tied directly to Pin 18 so the "execute" signal effectively causes the Model 2252A interface to begin an immediate transfer of data. The Up/Down position of the EXEC switch on the controller faceplate determines the signal level definition for the signal on Pin 31 (see Section 1.3).

> NOTE:
>
> Upon request, a Wang Service Representative can modify the Execute signal circuit on a Model 2252A interface controller to produce the following timing change. In the standard sequence, the Execute level is reset to logic "0" (not requesting) five microseconds after receipt of an input strobe; in the modified sequence, the Execute level is not reset to logic "0" until transfer of a readout to the CPU is completed. Such a change may be needed for devices designed to utilize a single signal level for two purposes: (1) as "request to send data", and (2) as "transfer-in-progress, indicating the current readout should not be changed ".

Pin 18

Input Strobe - - A 2 μs minimum pulse width input strobe or level transition from the interfaced device must be received on Pin 18 to initiate data transfer to the CPU via the Model 2252A controller, or Pin 31 must be tied to Pin 18 if the interfaced device is not designed to generate input strobes. The Up/Down position of the IS switch on the controller faceplate determines whether the leading edge of the strobe or level transition is low-to-high or vice versa (see Section 1.3).

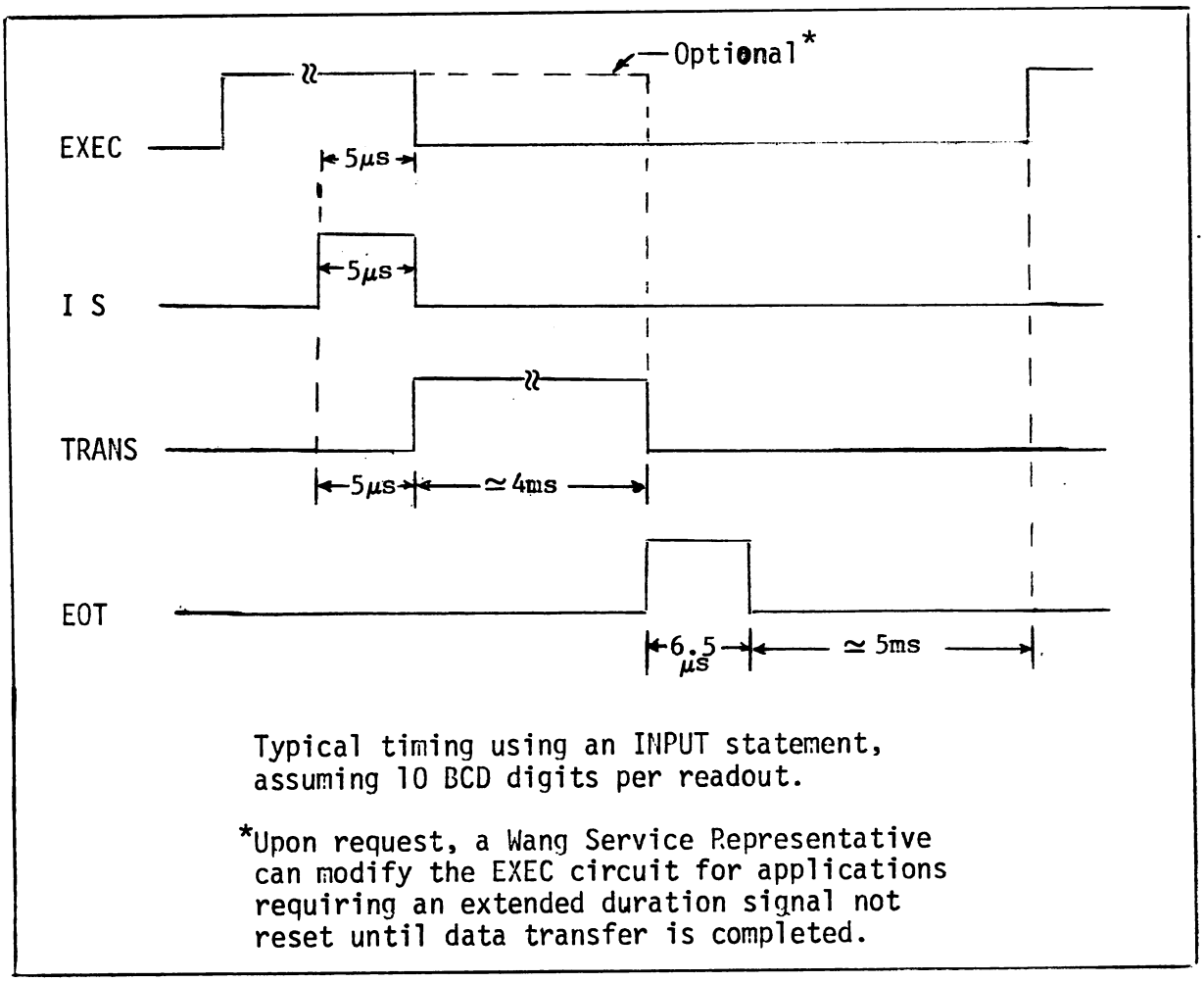
Pin 49

Transfer-in-progress Signal - - The outgoing "ready/busy" signal level on Pin 49 is set to logic "1" (busy, transferring) five microseconds after the Model 2252A interface receives an input strobe (or level transition) via Pin 18 and remains at the logic "1" level until all data in the readout has been transferred; then, the level is reset to logic "0" (ready, not transferring). If an interfaced instrument is designed to sense the level on Pin 49, a logic "1" indicates data transfer is in progress and the current readout should remain unchanged; a logic "0" indicates data transfer is not in progress and the readout can be changed. The Up/Down position of the TRANS switch on the controller faceplate determines the signal definition for the signal on Pin 49.

Pin 50 — EOT (End of Transfer) Output Strobe - - A 6.5μs pulse width output strobe is provided on Pin 50 at the time the Model 2252A interface transfers an ASCII HEX(0D) byte to the CPU denoting termination of the current readout transfer sequence. The Up/Down position of the EOT switch on the controller faceplate determines whether the leading edge of the strobe is low-to-high or vice versa (see Section 1.3).

Pin 32 — Prime Output Strobe - - A 5μs pulse width output strobe is generated when the RESET button on a Wang keyboard is depressed to interrupt processing and return system control to the operator. Generally, the signal is utilized by a device as a reset/initialization signal.

In Figure 1-1, a timing diagram is shown for the control signals Execute, Input Strobe, Transfer-in-progress, and End of Transfer. Several assumptions are indicated for the timing diagram.



Typical timing using an INPUT statement, assuming 10 BCD digits per readout.

*Upon request, a Wang Service Representative can modify the EXEC circuit for applications requiring an extended duration signal not reset until data transfer is completed.

Figure 1-1. Timing Diagram, All Switches Up

Typical I/O circuits for the Model 2252A Scanning Input Interface Controller are shown in Figures 1-2a and 1-2b.

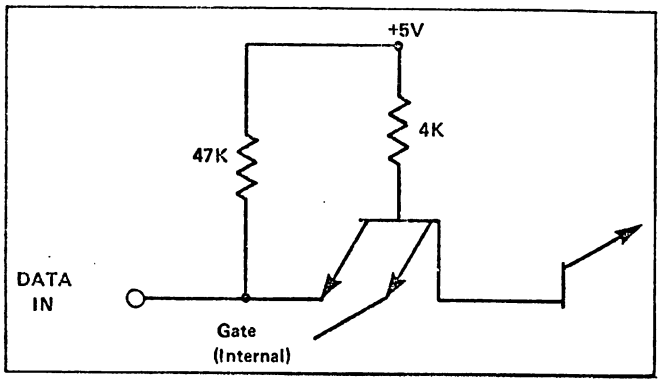


Figure 1-2a. Typical Input Circuit

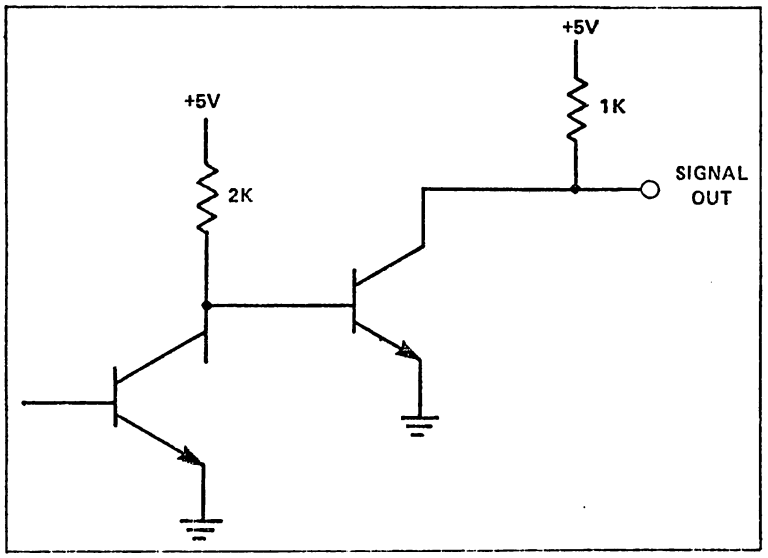


Figure 1-2b. Typical Output Circuit

1.3 LOGIC-LEVEL-SELECTION SWITCHES

Six "logic level selection" switches are located on the Model 2252A controller board faceplate (see Figure 1-3). The push button switches are provided because digital instrument designers do not use a standard convention when letting a pair of voltage levels ("low" and "high") represent a two-state condition such as "true" or "false", "on" or "off", and logic "0" or logic "1". Some designers let a low-signal denote logic "0" and a high-signal denote logic "1"; other designers reverse the definition.

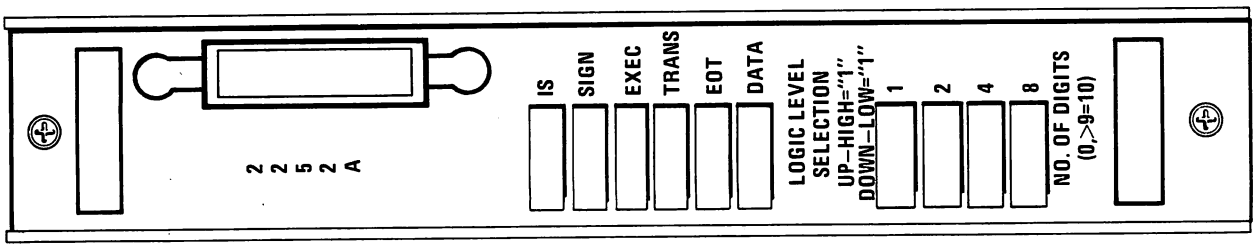


Figure 1-3. Logic-Level-Selection Switches

The proper Up/Down setting for each logic-level-selection switch on a Model 2252A controller board depends upon the function of a particular switch relative to the manufacturer's specifications for the particular device or instrument plugged into the controller.

NOTE:

1. Responsibility for setting the six logic-level-selection switches on the Model 2252A controller board should be assumed by the person who wires the Model 2252A male connector to the cable from a device being interfaced to the System 2200.

2. Once properly set, changes in the Up/Down positions of the six logic-level-selection switches should be unnecessary until a different instrument is plugged into the Model 2252A interface.

Guidelines for setting the logic-level-selection switches are given now. The guidelines may be omitted by any reader not responsible for setting the switches.

☐ SIGN The SIGN switch controls the signal level definition for the sign-bit (Pin 17) in each readout from the interfaced device. For processing by the System 2200, a plus (+) sign should be represented by logic "1" and a minus (-) sign by logic "0". Check the manufacturer's specifications; then set the SIGN switch as follows:

UP: If { + (logic "1") is the high-level signal, and
- (logic "0") is the low-level signal.

DOWN: If { - (logic "0") is the high-level signal, and
+ (logic "1") is the low-level signal.

☐ DATA The DATA switch controls the signal level definition for the up to forty data bits in each readout from the interfaced device. Check the manufacturer's specifications; then set the DATA switch as follows:

UP: If { logic "1" is the high-level signal, and
logic "0" is the low-level signal.

DOWN: If { logic "0" is the high-level signal, and
logic "1" is the low-level signal.

☐ EXEC The EXEC (Execute) switch controls the signal level definition for the outgoing logic "1" (request or initiation) level on Pin 31 of the connector. If the interfaced device requires a "request" level to initiate a readout or a settling condition, check the manufacturer's specifications; then set the EXEC switch as follows:

UP: If a low-to-high leading edge transition represents a "request" signal.

DOWN: If a high-to-low leading edge transition represents a "request" signal.

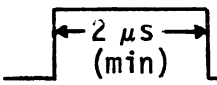
NOTE:

If Pin 31 has been tied to Pin 18 because the interfaced device cannot utilize the Execute (request) signal, the EXEC switch can be set UP or DOWN (but then the IS switch must be set identically).

☐ IS

The IS (Input Strobe) switch controls the definition of the input strobe (or the level transition) sent from the interfaced device to Pin 18 of the connector to initiate data transfer to the System 2200. If the device has an input strobe capability, check the manufacturer's specifications; then set the IS switch as follows:

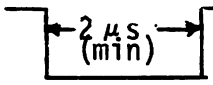
UP: If the leading edge of the strobe or level transition is low-to-high, as shown.



+2.5 vdc (min)

+0.4 vdc (max)

DOWN: If the leading edge of the strobe or level transition is high-to-low, as shown.



+2.5 vdc (min)

+0.4 vdc (max)

> NOTE:
>
> If the interfaced device does not have an input strobe capability and Pin 31 has been tied to Pin 18, the IS switch must be set the same as the EXEC switch; that is, both UP, or both DOWN.

☐ TRANS

The TRANS (Transfer-in-progress, busy) switch controls the signal level definition of the outgoing signal on Pin 49. The level is set to logic "1" (transferring, busy) five microseconds after the interface receives an input strobe (or level transition) via Pin 18. A logic "1" indicates data transfer is in progress and the current readout should remain unchanged. A logic "0" indicates data transfer is not in progress and the readout can be changed. If the interfaced device can utilize such a signal, check the manufacturer's specifications; then set the TRANS switch as follows:

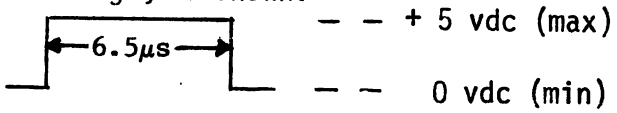
UP: If { logic "1" is the high-level signal, and
logic "0" is the low-level signal.

DOWN: If { logic "0" is the high-level signal, and
logic "1" is the low-level signal.

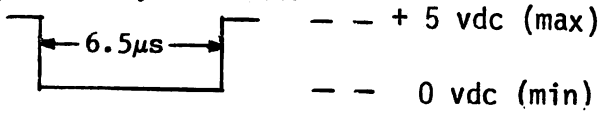
EOT

The EOT (End of Transfer) switch controls the definition of the 6.5 microsecond output strobe on Pin 50. The strobe indicates transfer of the current readout is complete. If the interfaced device can utilize such a strobe, check the manufacturer's specifications; then set the EOT switch as follows:

UP: If the leading edge of the EOT strobe should be low-to-high, as shown.



DOWN: If the leading edge of the EOT strobe should be high-to-low, as shown.



1.4 NUMBER-OF-DIGITS SWITCHES

Four "number of digits" switches are located on the Model 2252A controller board faceplace (see Figure 1-4). The switches are labeled 1, 2, 4, and 8, respectively. The switches can be set to indicate the exact number of BCD digits, any number from 1 to 10, to be strobed into the CPU for each readout. Also, for applications involving input of discrete binary data, the switches can be set to indicate the number of groups of discrete binary data (with 4-bits per group).

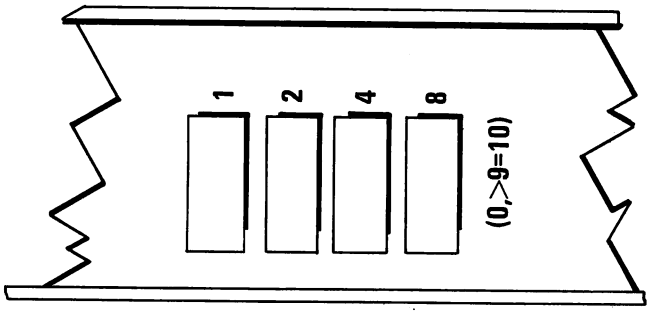


Figure 1-4. Number-of-Digits Switches

The labels on the number-of-digits switches are similar to position coefficients by which 4-bit binary numbers can be converted into decimal numbers. If a switch is Up, its position coefficient is multiplied by "zero"; if a switch is Down, its position coefficient is multiplied by "one". Now, if the position coefficients for all Down switches are summed, the sum can range from a minimum value of zero to a maximum value of fifteen; however, a zero sum or a sum greater than nine is converted automatically to the value ten by the interface. For example, if the number-of-digits to be strobed into the CPU for each readout is five, set the switches as follows: Down (4 and 1), Up (8 and 2).

NOTE:

1. Responsibility for setting the four number-of-digits switches on the Model 2252A controller board should be assumed by the person who wires the Model 2252A male connector to the cable from a device being interfaced to the System 2200.

2. Once properly set, changes in the Up/Down positions of the four number-of-digits switches should be unnecessary until a different instrument is plugged into the Model 2252A interface.

The number-of-digits selectivity feature of the Model 2252A interface provides the following advantages:

1. Wiring of unused BCD digit input circuits to a logic "0" level is not necessary.

2. Less time is required to transfer a readout to the CPU (approximately 0.5 milliseconds per BCD digit during INPUT statement execution).

3. Less memory is required when storing multireadouts in alphameric arrays during execution of DATALOAD BT, MAT INPUT, or $GIO statements.

If the setting on the number-of-digits switches is zero or a value greater than nine, ten BCD digits are read by the interface (in addition to the sign bit). The sign bit is read first; then, the BCD digits are read in the sequential order D_0, D_1, D_2, . . . , D_9. D_0 corresponds to the input circuits on Pins 1 through 4 in the connector. D_1 corresponds to the circuits on Pins 19 through 22. The correspondence for each BCD digit is defined in Table 1-1.

If the setting on the number-of-digits switches is "k", where $1 \leq k \leq 10$, the following BCD digits are read sequentially (after reading the sign bit); D_0, D_1, D_2, . . ., D_{k-1}.

1.5 SINGLE AND MULTI-UNIT DEVICE ADDRESSES

For program control of I/O operations, the System 2200 utilizes a three-hexadecimal-digit (12-bit) device addressing procedure with codes of the form xyy, where

x represents the Device Type, and

yy represents the Preset Address of a specific controller board (or one channel of a dual-channel board).

During execution of a particular I/O operation, the x-digit (that is, the four high-order bits in a 12-bit address code) determines which microcode programming routines are used by the system. The yy-digits (that is, the eight low-order bits in a 12-bit address code) determine which I/O device controller in the CPU chassis is enabled for execution of an operation.

The correct yy-digits for a particular I/O controller board correspond to the setting (the ON-OFF configuration) of an 8-pole address switch on the printed circuit (PC) board. The address switch is set before a board is shipped from the factory or when a board is installed in the CPU (see Appendix E). Care must be exercised, when an address switch is set, to ensure address uniqueness with respect to other boards installed in the same CPU chassis. Address settings should conform to the list of standard addresses recommended by Wang Laboratories (see Appendix A).

An address switch is neither visible nor accessible after a controller board is mounted in the CPU chassis of a System 2200 configuration. However, after an address switch on a PC board is set, a label showing the device address (or addresses, in the case of a dual-channel board) is attached to the faceplate of the controller board. Because the System 2200 BASIC language syntax for address codes requires a three-hexdigit address code, device address labels always show three-hexadecimal-digit codes which include a "standard" device type digit as well as the two hexdigits corresponding to the preset address of the particular board.

The device type digit in the address code shown on a controller board label is not always appropriate for every I/O operation to be performed by the controller board. Sometimes a different device type code must be used. If so, the recommended device type code for a particular operation is given usually with the general form of the I/O operation, or in a discussion of programming techniques for a particular controller board. On the other hand, the preset address digits in the address code shown on a controller board label must not be changed unless the address switch on the controller board is reset by a Wang Service Representative.

In Appendix A, the six standard address codes reserved for Model 2252A units are shown. The address codes are 25A, 25B, 25C, 25D, 25E, and 25F. In System 2200 configurations with only one Model 2252A interface, the device address of the interface should be 25A (that is, the preset address of the controller should be 5A). In a dual-unit configuration, the device address of one Model 2252A unit should be 25A and the address of the other unit should be 25B. If a third Model 2252A interface is added to a System 2200 configuration, the address of the third interface should be 25C.

Usually, instructions for reading and setting a Model 2252A address switch are unnecessary. However, if the person responsible for wiring the Model 2252A male connector to the cable from a non-Wang device needs to check the address switch on a Model 2252A interface, Appendix E contains instructions for setting the Model 2252A address switch.

CHAPTER 2

PROGRAMMING TECHNIQUES

2.0 INTRODUCTION

Some programming techniques for Model 2252A applications are presented in this chapter. However, if the chapter is to be completely meaningful, a reader must be familiar with the System 2200 BASIC language and with general programming techniques for the system.

2.1 DEVICE SELECTION

When the System 2200 is Master Initialized (that is, power is turned off and then on again), the "primary devices" are selected automatically for classes (groups) of input/output operations identified by the I/O-class parameters CI (console input), INPUT, CO (console output), PRINT, LIST, TAPE, DISK, and PLOT. To be a primary device, a device must be plugged into a controller board whose address is one of the five default addresses for the System 2200 (001, 005, 10A, 310, and 413).

For example, after Master Initialization, the keyboard controller with address 001 is selected automatically (that is, by default) for execution of all operations included in the I/O classes denoted by two parameters (CI and INPUT). Similarly, the CRT Executive Display controller with address 005 is selected automatically for execution of all operations included in the I/O classes denoted by three parameters (CO, PRINT, and LIST).

See Appendix B of this manual. There, the primary devices are listed in Table B-1. The I/O operations and class parameters for the System 2200 are identified in Chart B-1. Also, the general format of the SELECT statement is discussed since SELECT statements are used to assign addresses of nonprimary devices to I/O-class parameters. A study of Chart B-1 shows that some I/O-classes are identified by a parameter which is the same as the name of one operation included in the class (see INPUT, PRINT, LIST, and PLOT).

Any device or instrument interfaced to the System 2200 via a Model 2252A Scanning Input Interface Controller is not a primary device; that is, the Model 2252A controller is not selected by default for execution of any operation after the system is Master Initialized. Before a Model 2252A interface can be accessed by the system for execution of a particular operation, one of the following conditions must exist:

1. the address of the interface (for example, 25A) must be specified in the particular BASIC language statement being executed (if the statement syntax permits an address specification), or

2. the address of the interface must be the last address assigned, by a SELECT statement, to the I/O-class parameter governing the particular operation.

For example, upon execution, the statement

```
SELECT INPUT 25A
```

assigns the address 25A to the I/O-class parameter INPUT which governs device selection for execution of INPUT, KEYIN, and MAT INPUT statements.

Three sample System 2200/Model 2252A configurations are shown in Figure 2-1. Two configurations contain one Model 2252A interface; the third configuration contains two Model 2252A interfaces.

Now, assume the digital voltmeter in the dual-unit configuration is connected to the System 2200 via a Model 2252A interface with address 25A, and the digital clock is connected to the system via an interface with address 25B. Additionally, assume the following:

1. the system has just been Master Initialized,
2. a program stored on a tape cassette has been loaded into memory, and
3. the command RUN EXECUTE has been entered via the keyboard.

Furthermore, assume the first statement in the program now in memory is:

```
10 INPUT X
```

If so, the system recognizes the default address 001 (the keyboard) as the device to access for data input requested by the INPUT statement in Line 10. Consequently, a question mark appears on the CRT to indicate the system is awaiting input via the keyboard controller. However, if the program's objective is obtaining a readout from the digital clock, the program should begin as follows:

```
10 SELECT INPUT 25B
20 INPUT X
```

(See Example 2-1 for an illustration of how readouts are obtained from two instruments.)

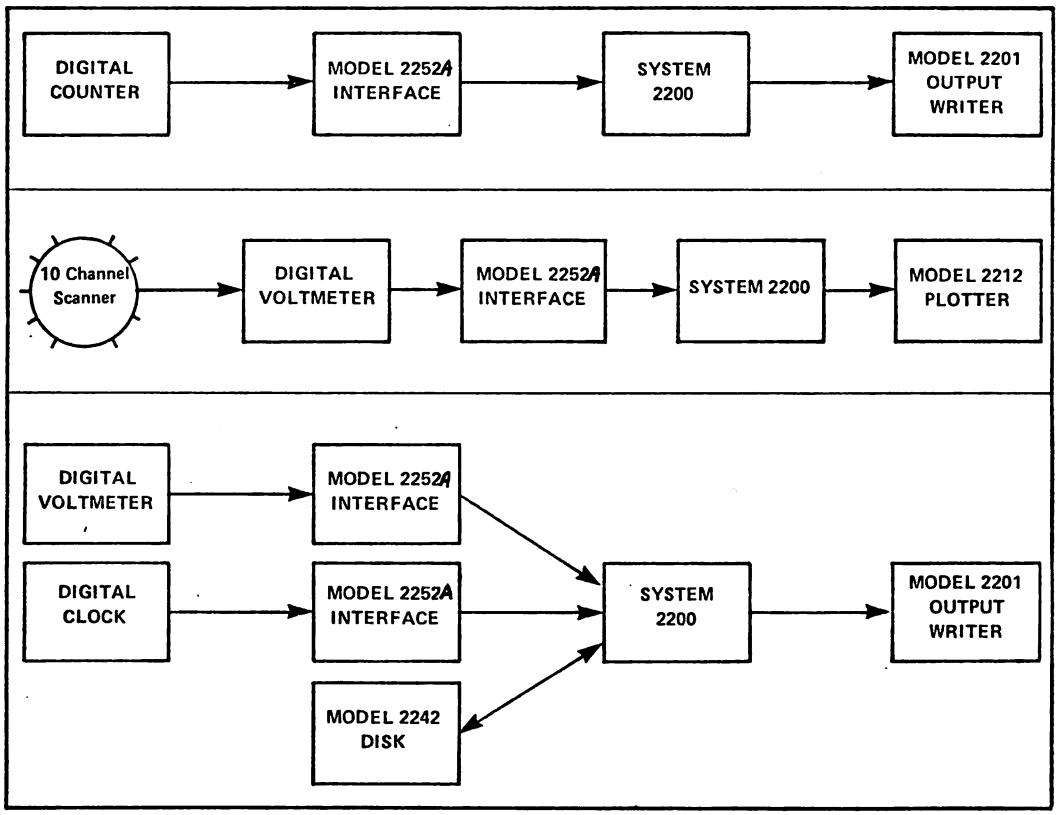


Figure 2-1. Sample System 2200/Model 2252A Configurations

Example 2-1. Requesting Readouts from Two Instruments

The following program uses a J-loop to request 50 readouts from an instrument connected to the System 2200 via an interface with address 25A. After each readout is received, the accumulated sum of the readouts is calculated; then the loop-counter, the corresponding readout, and the current sum are printed. After the last readout is received, the program requests one readout from an instrument connected to the system via an interface with address 25B. Finally, the program reassigns the primary devices to the I/O-class parameters INPUT and PRINT.

Program Sequence	Comments
10 S=0	Set sum to zero.
20 SELECT INPUT 25A, PRINT 211	Select interface 25A and the Output Writer.
30 FOR J=1 TO 50	Set up loop for 50 readouts.
40 INPUT X	Request one readout.
50 S=S+X	Accumulate sum of readouts.
60 PRINT "J=";J,"X=";X,"S=";S	Print loop-counter, readout, and sum.
70 NEXT J	Continue loop.
80 SELECT INPUT 25B	Select interface 25B.
90 INPUT Y	Request one readout.
100 PRINT "Y=";Y	Print the readout.
110 SELECT INPUT 001, PRINT 005	Reselect keyboard & CRT.

NOTE:

As a general programming practice, before a program terminates, primary devices should be reassigned to any I/O-class parameters to which nonprimary devices have been assigned. Otherwise, a particular nonprimary device currently selected for an I/O-class parameter may be inappropriate for the logic of another program, if the next program is run without first Master Initializing the system.

2.2 FIXED AND FLOATING POINT INPUT

To keep the Model 2252A logic as simple as possible, a decimal point character is not hardwired in the interface; data from an instrument is read into the CPU as an integer. Since an instrument usually is connected permanently to one interface, programming techniques similar to those presented in this section can be used to transform integer input into the appropriate fixed or floating point format.

After fixed point data is read into the system as an integer, the decimal point can be inserted in the data by using a scale factor in one program step. Table 2-1 shows several different fixed point formats in the left column. In the right column of the table, a two-line program sequence is shown for each fixed point format. Assuming the interface controller has been selected in an earlier step, a readout is requested by Line 50, and the integer value is stored in the numeric variable Y. Then, the decimal point is inserted in the value by Line 60.

Table 2-1. Sample Scale Factors for Fixed Point Readouts

Fixed Point Format*	Program Sequence
sxxxxxxxxx.x	50 INPUT Y 60 Y = .1*Y
sxxxxxxxx.xx	50 INPUT Y 60 Y = .01*Y
sxxxxx.xxxxx	50 INPUT Y 60 Y = Y*1E-5
s.xxxxxxxxxx	50 INPUT Y 60 Y = Y*1E-10

* s = sign; x = any BCD digit (where the number of digits read into the system depends upon the setting on the number-of-digits switches); the decimal point is not read into the system.

Numbers whose common notation consists of a fixed point quantity multiplied by an integral power of ten (e.g., 1.5923×10^6 or $.15923 \times 10^7$) are said to be in "scientific notation" if the decimal point appears after the first non-zero digit, and to be in "normalized notation" if the decimal point appears in front of the first non-zero digit. Similarly, the floating point representation of a number consists of two sets of digits: (1) the significant digits (non-zero first digit) in the number and (2) the exponential digits. The representation is said to be in normalized format if the implied decimal point is at the extreme left of the significant digits. For example, the normalized floating point format for the above numbers is 15923+07.

For Model 2252A applications, floating point formats can be read into the system as an integer. Then programming techniques similar to those shown in Examples 2-2 and 2-3 can be used to separate the exponential and significant digits in the data. However, scale factors (needed for separation of the two types of digits) cannot be chosen until after answers to the following questions are supplied by the person who wired the Model 2252A connector to the cable from the interfaced device:

1. What is the setting on the number-of-digits switches on the interface (that is, how many digits are transferred to the CPU per readout)? The answer can be any number from 1 to 10.

2. How many digits in each readout represent the exponent? The answer depends upon the design of the interfaced device.

3. Is an additional scale factor implied in each readout? The answer depends upon the design of the interfaced device.

4. Is the connector wired to receive the exponential digits in the leftmost (most significant) positions of the integer readout, or the rightmost (least significant) positions?

The following assumptions are made in Examples 2-2 and 2-3:

1. The setting on the number-of-digits switches is 8.

2. Each readout contains two exponential digits (therefore, six significant digits).

3. There is no built-in scale factor in the readouts.

4. In Example 2-2, the exponential digits are the rightmost digits in the readout; in Example 2-3, the exponential digits are the leftmost digits in the readout.

5. The sample readout used in the examples is equivalent to the normalized value $.756029 \times 10^4$ which is read into the system as 75602904 in Example 2-2, and as 04756029 in Example 2-3.

Example 2-2. Separating Exponential Digits Wired as Least Significant Readout Digits

Program Sequence	Comments
10 SELECT INPUT 25A	Select interface 25A for input.
20 INPUT X	Request a readout (e.g., 75602904).
30 X1 = X*.01	Insert a decimal point between the significant and the exponential digits (X1 = 756029.04).
40 X2 = INT(X1)	Separate the significant digits (X2 = 756029).
50 X3 = (X1-X2)*100	Separate the exponential digits [X3 = (756029.04 - 756029)*100 = 4].

Additional program logic can be written, using the significant digits now stored in X2, and the exponent stored in X3.

Example 2-3. Separating Exponential Digits Wired as Most Significant Readout Digits

Program Sequence	Comments
10 SELECT INPUT 25A	Select interface 25A for input.
20 INPUT X	Request a readout (e.g., 04756029).
30 X1 = X*1E-6	Insert a decimal point between the significant and the exponential digits (X1 = 4.756029).
40 X3 = INT(X1)	Separate the exponential digits (X3 = 4).
50 X2 = (X1-X3)*1E6	Separate the significant digits [X2 = $(4.756029-4)*10^6$ =756029].

As in Example 2-2, the significant digits are stored in X2, and the exponent is stored in X3. Additional program logic can be written, as required, for a particular application.

2.3 ERROR DETECTION FOR ILLEGAL BCD INPUT

Only ten of the sixteen unique 4-bit codes shown in Table 2-2 are legal input when data is read into the System 2200 via a Model 2252A interface during execution of an INPUT statement with a numeric argument; for example,

```
100 INPUT X
250 INPUT Y5
```

Table 2-2. Legal and Illegal Readout Codes for INPUT Statements with Numeric Arguments

	8-4-2-1 Codes	Equivalent BCD-digit	Equivalent Hexdigit
Legal	0000	0	0
	0001	1	1
	0010	2	2
	0011	3	3
	0100	4	4
	0101	5	5
	0110	6	6
	0111	7	7
	1000	8	8
	1001	9	9
Illegal	1010	None	A
	1011	None	B
	1100	None	C
	1101	None	D
	1110	None	E
	1111	None	F

The INPUT statement processing procedure includes one event not found in other input operations. When each byte is transferred to the CPU buffer from the controller board currently selected for INPUT-class operations, an "echo" of the byte is sent to the CRT display (unless another device has been selected for CO-class operations).

Once a carriage-return character is transferred to the CPU buffer, the system begins processing the data. A special event occurs if data is to be stored in a numeric variable location - - a test for legal characters is made before storing the data. If an illegal character is encountered (see Table 2-2), transfer of the buffered data is not made; an error message is sent to the CO (console output) device. The message appears on the line below the data echo. Next, the system requests additional data, repeatedly if necessary, until legal data is received for storage.

On the other hand, if data is to be stored in an alphameric variable (as required for applications involving discrete binary data), no test is made before storing the data - - all sixteen codes in Table 2-2 are legal for INPUT statements with alphameric arguments. However, even though no automatic error detection occurs in the processing procedure, an error detection procedure can be programmed to check data for numeric validity after its storage in an alphameric variable. See Section 2.5.

2.4 SUPPRESSING THE CRT INPUT-DATA-ECHO

When data is received by the System 2200 during execution of an INPUT statement, an echo of the data appears automatically as output on the device currently selected for console output (CO-class) operations. The echo usually appears on the CRT screen since the default address for CO-class operations is 005.

The INPUT-data-echo can be suppressed, if desired. For example, if an application requires the CRT display to remain undisturbed while readouts are being received from an interfaced device, use a programming technique similar to the one in Example 2-4.

Select the address of the Model 2252A interface for console output operations immediately before each single INPUT statement appearing in a program or immediately before each multi-readout INPUT-statement-loop (i.e., a loop containing only an INPUT operation, as illustrated in Example 2-4). Then, the INPUT-data-echo is sent to the interface rather than the CRT; but, because the interface is an input-only device, the echo characters are ignored. However, the CRT must be reselected for console output operations immediately after each single INPUT statement appearing in a program or immediately after each multi-readout INPUT-statement-loop appearing in a program. Reselection of the CRT for console output operations is necessary because some system operations included in the CO-parameter category can cause trouble if implemented via a Model 2252A interface controller which has been left selected while additional program logic is being executed.

Alternatively, select the address of the Line Printer (usually 215) or the Output Writer (usually 211) for console operations, thereby obtaining a hard copy of the INPUT-data-echo.

Example 2-4. Suppressing the CRT INPUT-data-echo

Program Sequence	Comments
10 DIM Y(100)	Dimension the Y-array for 100 elements.
20 SELECT INPUT 25A	Select interface 25A for INPUT-class operations.
.	
.	
.	
80 SELECT CO 25A	Send echo to interface during INPUT loop.
90 FOR J=1 TO 100	Set up the J-loop for 100 readouts.
100 INPUT Y(J)	Request a readout for the Jth element of Y().
110 NEXT J	Continue the loop.
120 SELECT CO 005	Reselect CRT for CO-operations.
.	
.	
.	
200 SELECT INPUT 001	Reselect the keyboard.

NOTE:

Do not attempt to send the INPUT-data-echo to any device controller address not included in the CPU chassis of the system being used. If a nonexistent address is assigned to the CO-parameter in a SELECT statement, the system locks out when the statement is executed. To regain control, the operator must Master Initialize the system, thereby clearing all memory.

2.5 PROGRAM CONTROL OF INPUT ERRORS

Program control of erroneous BCD input is possible using NUM functions and CONVERT statements. The NUM function and the CONVERT statement are available in the System 2200 B, C, S and T (but not in the System 2200A).

Briefly, a NUM function determines the number of valid numeric bytes, including sign bytes, stored in a specified alphameric variable. The NUM function can be included in an IF...THEN statement to test a specified condition.

A CONVERT statement converts the numeric value stored in a specified alphameric variable from the ASCII format (that is, 8-bits per decimal digit and per sign) to the System 2200 internal numeric format (that is, 4-bits per decimal digit and 1 bit per sign) and then stores the result in a specified numeric variable.

In Example 2-5, an INPUT statement with an alphameric argument is used to request a readout (Line 110). The readout stored in the alphameric variable is tested for numeric validity (Line 120) by comparing the number of valid numeric bytes in X$ with the total number of bytes in X$ (determined by the LEN function). The LEN (length) function is very useful for Model 2252A applications since it calculates the total number of bytes in a specified variable, thereby removing the necessity to specify the number of bytes.

After the test is made in Line 120, a conditional branch to Line 200 is made if X$ contains any non-numeric data. No illustrative programming logic is given for Line 200; however, processing of erroneous values might consist of printing operations or counting operations. If X$ contains only numeric data, no branch is made to Line 200; program execution proceeds from Line 120 to Line 130, where the system is instructed to convert the data for storage in the numeric variable X.

Example 2-5. Processing Input Errors

```
      Program Sequence                    Comments

100 SELECT INPUT 25A                  Select the Model 2252A interface for
                                      input.
110 INPUT X$                          Request a readout and store in X$.
120 IF NUM(X$)<LEN(X$) THEN 200       If X$ contains any non-numeric bytes,
                                      branch to the logic for processing
                                      input errors.
130 CONVERT X$ TO X                   Convert the readout to an internal
 .                                    numeric value and store in X.
 .
 .
190 GOTO 300                          Continue execution at Line 300.
200 REM PROCESS ERROR                 Remark indicating start of error
 .                                    processing logic.
 .
 .
300 SELECT INPUT 001                  Reselect keyboard.
```

A substantially longer program sequence is needed to accomplish the same results if a System 2200A is used since the NUM function and the CONVERT statement used in Example 2-5 are not available in the System 2200A.

2.6 DISCRETE BINARY INPUT

Up to 41 discrete binary bits can be supplied, in parallel, to the Model 2252A interface for storage in an alphameric variable (or array element) specified in an INPUT statement. With an alphameric argument, the automatic error detection procedure discussed in Section 2.3 is bypassed, thereby ensuring the acceptability of any set of contiguous binary bits.

Discrete binary input is read by the interface in the sequential order described in Section 1.4 and converted into ASCII codes as described in Section 1.2. For example, a discrete bit received via Pin 17 of the interface connector is transferred to the CPU as the low-order bit in a HEX(2B) or HEX(2D) code which is stored as the first byte in the specified alphameric variable. Four discrete bits received via Pins 1 through 4 become the 1-2-4-8 low-order bits, respectively, in a HEX(3h) code; that is, the discrete bits determine the low-order hexdigit h. The resulting 8-bit code is stored as the second byte in the alphameric variable. Similarly, four discrete bits received via Pins 19 through 22 are converted to another HEX(3h) code and stored as the third byte in the alphameric variable. The conversion and transfer procedure continues until the number of 4-bit-groups of discrete binary data equals the setting on the number-of-digits switches.

After discrete binary input is stored in memory, the data can be processed by straightforward programming techniques such as those illustrated in Examples 2-6 through 2-9. Usually, programming logic for discrete binary applications can be simplified by using one or more bit and byte manipulation statements or functions shown in Table 2-3. The table indicates statement/function availability with respect to four System 2200 CPU models. Also, the data conversion statements $TRAN, $PACK, and $UNPACK in the General I/O ROM (Read Only Memory) can be used to process discrete binary data.

Table 2-3. Bit and Byte Manipulation Statements/Functions (Availability denoted by "x")

Statement/ Function	2200A	2200B, C & T	2200S	2200S with Opt. 22, 23 or 24
ADD		x		x
AND,OR,XOR		x		x
BIN		x		x
BOOL		x		x
CONVERT		x	x	x
HEXPRINT		x	x	x
INIT		x		x
NUM		x	x	x
PACK		x		x
POS		x		x
ROTATE		x		x
UNPACK		x		x
VAL		x	x	x

Example 2-6. Isolating and Testing a Particular Bit in Discrete Data

Assume a Model 2252A interface with address 25B is supplied with 13 bits in parallel, on input circuits corresponding to the sign bit and the BCD digits denoted by D_0, D_1, and D_2 in Table 1-1. The following program sequence stores the data in the variable Y$ and then tests the data to determine whether the low-order bit corresponding to D_1 is equal to "1".

```
      Program Sequence                    Comments

10  DIM Y$4, B$1                  Dimension Y$ and B$.
20  SELECT INPUT 25B              Select the interface.
30  INPUT Y$                      Request a readout and store in Y$.
40  B$ = STR(Y$,3,1)              Store the 3rd byte of Y$ in B$.
50  AND(B$,01)                    Replace the seven high-order bits in B$ by zeroes.
60  IF B$ = HEX(01) THEN 150      Test for low-order bit = 1.
```

Now, consider the following application. A Model 2252A interface is supplied with a readout representing a sign and a 16-bit binary value. Receive and store the readout; then convert the binary value to an equivalent decimal system value.

Programming techniques for such an application are presented in Examples 2-7 through 2-9. Example 2-9 requires less execution time than Examples 2-7 and 2-8.

By assumption, in Examples 2-7 and 2-8, the Model 2252A connector is wired to receive a sign bit and a 16-bit binary value as shown in Figure 2-2. Therefore, to convert the binary value to its decimal equivalent, first convert each group of four binary bits to a hexdigit and then evaluate a polynomial in powers of 16. Setting, h_0 = the equivalent hexdigit for D_0, h_1 = the hexdigit for D_1, h_2 = the hexdigit for D_2, and h_3 = the hexdigit for D_3, then the algorithm for the absolute value of the decimal number is

$$X = \sum_{j=0}^{3} (16)^{3-j} h_j = (16)^3 h_0 + (16)^2 h_1 + 16h_2 + h_3$$

See Lines 60 through 90 in Example 2-7 and Lines 50 through 80 in Example 2-8 for some programming techniques utilizing the algorithm.

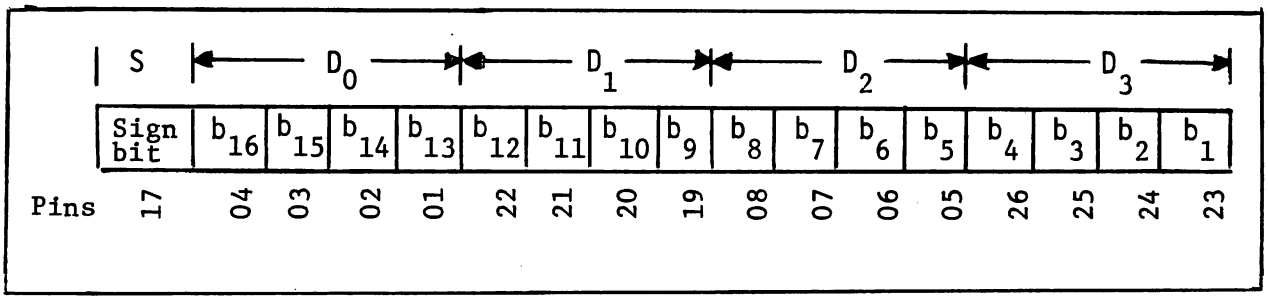


Figure 2-2. Binary Input Schematic for Examples 2-7 and 2-8

Example 2-7. Converting a Binary Value to its Decimal System Equivalent

Program Sequence	Comments
10 DIM X$5, B$1	Dimension variables X$ and B$.
20 SELECT INPUT 25A	Select the interface.
30 INPUT X$	Request a readout.
40 B$ = STR(X$,1,1)	Save the sign.
50 AND(X$,0F)	Replace the four high-order bits in each byte by zeroes.
60 X = VAL(STR(X$,5,1,))	Let X = floating point value (f.p.v.) of the fifth byte of X$.
70 X = X+16*VAL(STR(X$,4,1))	Add 16 times the f.p.v. of the fourth byte of X$.
80 X = X+256*VAL(STR(X$,3,1))	Add 256 times the f.p.v. of the third byte of X$.
90 X = X+4096*VAL(STR(X$,2,1))	Add 4096 times the f.p.v. of the second byte of X$.
100 IF B$ = "+" THEN 120	If sign positive, go to Line 120
110 X = -X	Correct the sign.
120 REM NEXT LOGIC	

Example 2-8. A Variation of Example 2-7

Program Sequence	Comments
10 DIM X$5	
20 SELECT INPUT 25A	
30 INPUT X$	
40 AND(X$,0F)	
50 X = 0	
60 FOR J = 2 TO 5	Set up J-loop to evaluate the floating point value.
70 X = 16*X + VAL(STR(X$,J,1))	
80 NEXT J	
90 X = X*(12-VAL(X$))	Multiply by plus or minus one.

In Line 90 of Example 2-8, the function VAL(X$) calculates the floating point equivalent of the first byte of X$. In Line 40, the first byte of X$ is replaced by HEX(0B) or HEX(0D), depending on the original input. Therefore, 12-VAL(X$) is equal to plus one for $(0B)_{16}$ or minus one for $(0D)_{16}$. See Example 2-9 for a more rapid conversion method.

Example 2-9. Reducing Execution Time for Binary to Decimal Conversion

By wiring the Model 2252A connector to receive a sign bit and a 16-bit binary value as shown in Figure 2-3, and using the ROTATE statement, less execution time is required for binary to decimal conversion than the time required for execution of the methods in Examples 2-7 and 2-8. By comparison with Figure 2-2, observe that the 4 highest-order bits are received on the D_0 input circuits, and the 4 lowest-order bits are received on the D_3 input circuits in both diagrams; the bits corresponding to hexdigit h_1 are received on the D_2 circuits, and the bits corresponding to hexdigit h_2 are received on the D_1 circuits in Figure 2-3 (in contrast to the wiring in Figure 2-2).

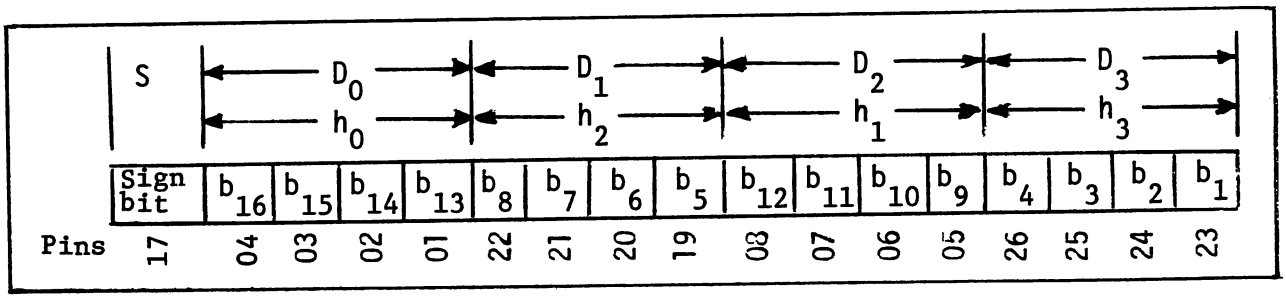


Figure 2-3. Binary Input Schematic for Example 2-9

Program Sequence	Comments
10 DIM X$5	
20 SELECT INPUT 25A	
30 INPUT X$	
40 AND(X$,0F)	
50 ROTATE(STR(X$,2,2),4)	
60 OR(STR(X$,2,2),STR(X$,4,2,))	
70 X = 256*VAL(STR(X$,2,1))+VAL(STR(X$,3,1))	
80 X = X*(12-VAL(X$))	

X$	$(2B)_{16}$ or $(2D)_{16}$	$(3h_0)_{16}$	$(3h_2)_{16}$	$(3h_1)_{16}$	$(3h_3)_{16}$
X$	$(0B)_{16}$ or $(0D)_{16}$	$(0h_0)_{16}$	$(0h_2)_{16}$	$(0h_1)_{16}$	$(0h_3)_{16}$
X$	$(0B)_{16}$ or $(0D)_{16}$	$(h_00)_{16}$	$(h_20)_{16}$	$(0h_1)_{16}$	$(0h_3)_{16}$
X$	$(0B)_{16}$ or $(0D)_{16}$	$(h_0h_1)_{16}$	$(h_2h_3)_{16}$	$(0h_1)_{16}$	$(0h_3)_{16}$

2.7 SCANNING OPERATIONS

The scanning feature of the Model 2252A interface can offer advantages for applications involving devices or instruments with long settling or response times.

Programming techniques for scanning operations utilize one of the following BASIC language statements:

1. KEYIN, available in the standard System 2200 B, C, S and T.

2. $IF ON, available in the standard System 2200T, in the System 2200 B and C with Option 2, and in the System 2200S with Option 23 or 24.

Neither statement is available in a System 2200A configuration.

In a typical scanning operation, a KEYIN or a $IF ON statement is used periodically to test the data-ready-condition of a Model 2252A interface. Between tests, other program logic can be executed or other devices scanned.

The scanning mode is initiated the first time a KEYIN or $IF ON statement is executed and remains in effect until data is received, regardless of how many times one particular statement is executed or several such statements are executed.

The general form of the KEYIN statement is:

KEYIN alpha-variable, line-number, line-number

According to the statement syntax, an alphameric variable (or array element) and two line numbers must be specified. For Model 2252A applications, the line numbers should be identical.

No address code can be specified in a KEYIN statement. During KEYIN execution, the system enables the device currently selected for INPUT-class operatiors, or the default device (the keyboard with address 001). Therefore, to scan an interfaced device with a KEYIN statement, the address of the interface controller must be assigned to the SELECT statement parameter INPUT, as shown in Examples 2-10 and 2-11.

When a Model 2252A interface is enabled for a KEYIN operation, the interface sets the Execute signal level on Pin 31 to logic "1" to indicate the system is ready to receive input from the interfaced device; the "request" level on Pin 31 remains set until input is received (see Section 1.2). Then, the CPU tests the data-ready-condition on the interface board and implements one of the following procedures:

1. If a data-not-ready-condition is sensed, the interface is disabled. Program execution advances to the next program statement. (The "request" level remains in effect on the interface board, and one readout can be received while the board is disabled.)

2. If a data-ready-condition is sensed, the interface reads the sign bit (the input signal on Pin 17), converts the bit to a HEX(2B) or HEX(2D) byte, and transfers the byte to the CPU. The sign byte is stored as the first byte in the specified alphameric argument, and program execution branches to the first specified line number.

When a data-ready-condition exists during a KEYIN scanning operation, the sign of a readout is received separately since only one byte is processed by a KEYIN statement. However, an INPUT statement can be used to receive the remaining information in the readout, as shown in Examples 2-10 and 2-11.

The general form of the $IF ON statement is:

$IF ON [#n, /xxx,] line-number

Though optional, address specification in a $IF ON statement is recommended. (If neither an absolute nor indirect address is specified, the system enables the controller board currently selected for TAPE-class operations.) One line number must be specified in an actual statement.

When a Model 2252A interface is enabled for a $IF ON operation, the CPU tests the interface data-ready-condition. If a not-ready condition is sensed, program execution advances to the next program statement. If a ready condition is sensed, program execution branches to the specified line number. No data is read.

Use of a $IF ON statement is shown in Example 2-12.

Example 2-10. Scanning with a KEYIN Statement

```
      Program Sequence                  Comments

10  DIM X$11                    Dimension X$ (one more than no.-of-digits
                                setting, assumed here to be 10).
20  SELECT INPUT 25A            Select interface 25A for INPUT-class
                                operations.
30  KEYIN X$, 100, 100          (On 1st pass, initiate scanning mode.
                                On subsequent passes, test data ready/not-ready.
                                If ready, store readout sign as 1st byte in
                                X$, and branch to Line 100 to receive
                                remaining bytes.
40  REM OTHER LOGIC             If not-ready in Line 30, begin other logic.
 .
 .
 .
90  GOTO 30                     Return to scanning operation.
100 INPUT STR(X$,2,10)          Receive and store rest of readout.
110 CONVERT X$ TO X             Convert to numeric value.
120 REM BEGIN NEXT LOGIC
```

Example 2-11. A Variation of Example 2-10

```
Program Sequence                    Comments

10  DIM X$1                         Dimension X$ (one byte).
20  SELECT INPUT 25A                Select interface 25A.
30  KEYIN X$, 100, 100              (See Line 30 in Example 2-10.)
40  REM OTHER LOGIC                 If not-ready, begin other logic.
 .
 .
90  GOTO 30                         Return to scanning operation.
100 INPUT X                         Receive readout without sign and store in X.
110 IF X$ = "+" THEN 130            If sign positive, skip to Line 130.
120 X = -X                          Insert correct sign in X.
130 REM BEGIN NEXT LOGIC
```

Example 2-12. Scanning with a $IF ON Statement

```
Program Sequence                    Comments

10  SELECT INPUT 25A                Select interface 25A
20  $IF ON /25A, 100                On 1st pass, initiate scanning mode.
                                    On subsequent passes, test data
                                    ready/not-ready.
                                    If ready, branch to Line 100.
30  REM OTHER LOGIC                 If not ready in Line 20, begin other logic.
 .
 .
90  GOTO 20                         Return to scanning operation.
100 INPUT X                         Receive readout and store in X.
110 REM BEGIN NEXT LOGIC
```

2.8 MULTI-READOUTS VIA SINGLE BASIC STATEMENTS

In previous sections, INPUT statements are used to request readouts via a Model 2252A interface; however, other BASIC language statements offer advantages for applications requiring a block of readouts to be stored before data processing begins. The statements are:

1. DATALOAD BT (device type = 6), available in the standard System 2200 B, C and T, and in the System 2200S with Option 22, 23 or 24.

2. MAT INPUT, available in the standard System 2200T, in the System 2200B and C with Option 1, and in the System 2200S with Option 21, 22, 23 or 24.

3. $GIO, available in the standard System 2200T, in the System 2200 B and C with Option 2, and in the System 2200S with Option 23 or 24.

Each statement provides the capability to receive a block of readouts for storage in a specified array. For DATALOAD BT and $GIO statements, the specified array must be alphameric; for MAT INPUT statements, the specified array can be numeric or alphameric.

In Table 2-4, "best case" comparisons are given in milliseconds per readout for the following sample program statements:

1. INPUT with numeric arguments:

```
INPUT A, B, C, D, E
```

2. INPUT with alphameric arguments:

```
INPUT A$, B$, C$, D$, E$
```

3. MAT INPUT with numeric array:

```
DIM A(10)
MAT INPUT A
```

4. MAT INPUT with alphameric array:

```
DIM A$(10)11
MAT INPUT A$
```

5. DATALOAD BT with alphameric array:

```
DIM A$(10)12
DATALOAD BT /65A, A$()
```

6. $GIO with alphameric array:

```
DIM A$(10)12, B$10
$GIO /25A (C640, B$) A$()
```

To each time given in Table 2-4 must be added any response time of the instrument and any additional delays due to different programming techniques. Also, in general, the time required to convert data stored in specified arguments (or to save data on another peripheral) is longer than the time required to read the data into memory initially.

Table 2-4. Best-Case Times in Milliseconds Per Readout*

Statement \ Number-of-digits	10	7	4	2	1
INPUT (numeric argument)	8.6	7.8	6.4	5.4	4.8
INPUT (alphameric arguments)	8.1	6.9	5.7	4.9	4.5
MAT INPUT (numeric array)	8.2	7.1	5.9	5.1	4.7
MAT INPUT (alphameric array)	7.8	6.7	5.5	4.8	4.4
DATALOAD BT /65A	1.1	0.84	0.56	0.37	0.28
$GIO /25A (C640	1.0	0.75	0.5	0.33	0.25

*For devices with settling times in excess of 10 milliseconds, percentage gains implied by values in the table are overshadowed by the slowness of the interfaced device.

During execution of a DATALOAD BT, MAT INPUT, or $GIO statement, the multi-readout transfer sequence usually terminates when the specified array has been filled. However, under certain circumstances, data transfer can be terminated by a special code if signal levels can be gated into the Model 2252A connector to produce a unique non-BCD digit code. When statement execution begins, the Model 2252A interface is enabled for the particular operation; the controller board sets the Execute level on Pin 31 to logic "1" to indicate the system is ready to receive input. Five microseconds after an input strobe is received by the interface, the Execute level is set to logic "0" by the interface and remains so while transfer of the readout to the CPU is in progress. When transfer of the current readout is complete, the CPU presents a ready level to the interface; the interface, in turn, sets the Execute level to logic "1" to request another readout. The process continues until sufficient readouts are received to satisfy the array dimensions, or until a special termination code is received, or until a predetermined number of readouts is received.

2.81 THE DATALOAD BT STATEMENT

The general form of the DATALOAD BT statement for Model 2252A applications is:

DATALOAD BT [([N=expression],[S={hh / alpha-variable-1}])] [#n, / /xyy,] {alpha-variable-2 / alpha-array-designator}

where, the components are defined as follows:

N	A parameter used to denote the number of bytes to be read.
expression	A specified number, or an expression to be evaluated (and truncated to an integer which must be ≥ 1).
S	A parameter used to denote a termination (stop) code.
hh	A pair of hexdigits specifying a termination code.
alpha-variable-1	An alphameric variable whose first byte specifies a termination code.
n	An indirect address, a file number (1,2,3,4,5, or 6).
xyy	An absolute address, whose hexdigit x must be 6 and hexdigits yy must be the preset address of the interface (e.g., 5A,5B,5C,5D,5E, or 5F).
alpha-variable-2	A specified alphameric variable or array element, defining a storage location.
alpha-array-designator	An entire array (all elements in a one or two dimensional array) defining a block of storage into which consecutive bytes are read. Elements of a two-dimensional array are filled row-by-row. An array designator is denoted by an array name followed by left and right parentheses; e.g., X$(), Y5$().

Note:

1. A comma must separate the N and S parameters if both are specified; e.g., (N=1200, S=3F).

2. If an array is specified, its dimensions must be specified previously in a DIM statement. Array dimensions are restricted as follows:

   a) bytes per element, 64 maximum,
   b) elements per row, 255 maximum,
   c) number of rows, 255 maximum, and
   d) elements per array (i.e., number of rows times elements per row), 4096 maximum.

3. Array dimensions must be large enough to accommodate the specified or expected number of bytes but not large enough to produce a table overflow condition.

The DATALOAD BT statement offers the following features:

1. A device address (device type = 6) can be specified, so no SELECT statement is necessary.

2. A predetermined number of readouts can be received and stored in an alphameric array.

3. A termination code can be specified.

During statement execution, readouts are accepted until one of the following conditions is satisfied (whichever occurs first):

1. The specified array is filled.

2. The specified number of bytes is received.

3. The specified termination code is received.

If execution is terminated by a specified code, the interface will not have completed its current readout transfer cycle which ends with a carriage-return character, i.e., a HEX(0D) byte. Therefore, an additional DATALOAD BT statement is needed to complete the cycle as shown in Example 2-15.

During execution, successive readouts are stored in successive elements of the specified array. The data in each readout are converted by the interface to ASCII character codes before storage, and the carriage-return character is stored also. (The carriage-return character is not stored during execution of an INPUT statement.) For convenience, the length of each array element should be equal to the setting on the number-of-digits switches plus one additional byte for the sign of the readout and one additional byte for the carriage-return character.

Example 2-13 illustrates the use of a DATALOAD BT statement to receive 100 readouts via a Model 2252A interface, assuming the setting on the number-of-digits switches is 10.

Example 2-13. Using a DATALOAD BT Statement

```
      Program Sequence                          Comments

10  DIM X$(100)12, X(100)              X$-array: 100 elements, 12 bytes/element.
20  DATALOAD BT /65A, X$()             Request 100 readouts for storage in X$().
30  FOR J = 1 TO 100                   Set up J-loop.
40  CONVERT STR(X$(J),1,11) TO X(J)    Convert each readout (excluding
                                       carriage-return byte) to a numeric value.
50  NEXT J
```

Example 2-14. Specifying N in a DATALOAD BT Statement

The parameter N can be used to specify the total number of bytes to be received in a DATALOAD BT operation. A variable or an integer can be assigned to the parameter N. If a variable is assigned, the system calculates the value of the variable and truncates the result to an integer. If the integer is not greater than or equal to one, execution is terminated and an error message shown on the CRT. A sample format to receive 20 readouts with 12 bytes per readout (assuming a 10 BCD digit readout) is:

```
10  DIM X$(100)12
20  DATALOAD BT (N=240) /65A, X$()
```

Example 2-15. Specifying S in a DATALOAD BT Statement

The parameter S can be used to specify a special termination code. The technique is useful for applications where the number of readouts cannot be predetermined. However, a termination signal level must be available in the interfaced device; the signal must be gated to the Model 2252A connector and must produce a unique non-numeric 4-bit code (see Table 2-2). In applications not requiring a sign as part of the readout, the Pin 17 level can be utilized for an S code. Since a logic "1" on Pin 17 produces a HEX(2B) code when a readout is transferred to the CPU, set S=2B as shown in Line 20 - - if a logic "1" from the device corresponds to a termination signal:

```
      Program Sequence                          Comments

10  DIM X$(100)12, R$12                For 10 BCD digits per readout.
20  DATALOAD BT (S=2B) /65A, X$()      Receive readouts until an ASCII plus is
                                       received.
30  DATALOAD BT (S=0D) /65A, R$        Complete transfer of last readout which
                                       terminates with a carriage return code.
```

Now, assume the connector on another interface (standard address 25B) is wired to produce a hexdigit F code when a termination signal is available. Since the code is converted to a HEX(3F) by the interface, set S=3F as shown in Line 120 below.

```
      Program Sequence                          Comments

110 DIM Y$(300)8, Z$8                  For 6 BCD digits per readout.
120 DATALOAD BT (S=3F) /65B, Y$()      Receive readouts until an incoming
                                       hexdigit F is received.
130 DATALOAD BT (S=0D) /65B, Z$        Complete transfer of last readout.
```

The "stop character" (the termination code) is stored in the specified alphameric array. However, as shown in the following sequence, the POS function can be used to isolate the array element containing the termination character while data is being converted. (Line 140 is unnecessary; the Y-array dimension should be included in Line 110.)

```
140 DIM Y(300)
150 J = 1
160 IF POS(Y$(J) = HEX(3F)) >0 THEN 190
170 CONVERT STR(Y$(J),1,7) TO Y(J)
180 J = J+1 : IF J<= 300 THEN 160
190 REM CONVERSION LOGIC COMPLETE
```

2.82 THE MAT INPUT STATEMENT

The syntax and features of the MAT INPUT statement are presented in detail in the Matrix Statements Reference Manual.

The MAT INPUT statement is similar to the INPUT statement with respect to a data echo feature. During execution of MAT INPUT statements, an echo of each received byte is sent to the device currently selected for CO (console output) operations.

The MAT INPUT statement is unlike the INPUT statement with respect to specification of an argument list. For an INPUT statement, acceptable arguments are limited to numeric or alphameric variables, including specific array elements; an array designator is not acceptable. For a MAT INPUT statement, the system automatically interprets each specified argument as an array (array designator notation is implied but not used) -- the syntax requires specification of numeric or alphameric array names without the left and right parenthesis used in the standard array designator notation. (A DIM statement must supply the dimensions of each array included in a MAT INPUT statement.) See Examples 2-16 through 2-19.

When compared with the DATALOAD BT statement, the MAT INPUT statement offers an advantage for numeric-only multi-readout applications since readouts can be received and stored directly in specified numeric arrays, thereby reducing subsequent data processing requirements. On the other hand, the transfer rate per readout for MAT INPUT statements is less favorable than the rate for DATALOAD BT statements. The MAT INPUT rate per readout is constrained by a 0.5 ms transfer rate per byte compared to a 0.1 ms transfer rate per byte for DATALOAD BT operations. Thus, there are trade-offs to consider when programming a particular application if both the DATALOAD BT and the MAT INPUT statements are available.

If an alphameric array is specified in a MAT INPUT statement, readouts received via the Model 2252A interface are stored as ASCII characters. If a numeric array is specified, readouts are converted to the System 2200 internal numeric format before storage.

The carriage-return character supplied by the interface to denote the end of each readout is not stored during a MAT INPUT operation (a feature similar to the INPUT statement). To conserve storage, the length of each alphameric array element should be equal to the setting on the number-of-digits switches plus one additional byte for the sign of the readout. Exact length specification is not required, however, since each array element is filled with space characters if a readout contains less bytes than the dimension of the element.

No device address can be specified in a MAT INPUT statement. During execution, the system enables the device currently selected for INPUT-class operations, or the default device (the keyboard with address 001).

Example 2-16. Using a MAT INPUT Statement

The following sequence receives 36 readouts via a Model 2252A interface with address 25A and stores the readouts in ASCII codes. Twenty readouts are stored in X$() and sixteen readouts in Y$(). Since the carriage-return character is not stored, the readouts can be converted as shown in Lines 40 through 60.

```
10  SELECT INPUT 25A
20  DIM X$(20)11, Y$(4,4)11, X(20)
30  MAT INPUT X$, Y$
40  FOR J = 1 TO 20
50  CONVERT X$(J) TO X(J)
60  NEXT J
```

Example 2-17. Direct Storage in a Numeric Array

The following sequence receives 30 readouts and converts each readout to a numeric format before storage.

```
10  SELECT INPUT 25A
20  DIM X(30)
30  MAT INPUT X
```

Example 2-18. Suppressing the CRT MAT INPUT Data Echo

The MAT INPUT data echo can be suppressed by the same technique used in Example 2-4 (see Section 2.4).

```
10  DIM C(30)
20  SELECT INPUT 25A, CO 25A
30  MAT INPUT C
40  SELECT INPUT 001, CO 005
```

Example 2-19. Redimensioning an Array in a MAT INPUT Statement

Another feature of MAT INPUT statements may be useful for Model 2252A applications. An array can be redimensioned in the statement. However, a two-dimensional array cannot be redimensioned as a one-dimensional array, and vice versa. During execution of a MAT INPUT statement, the last specified dimensions apply unless new dimensions, enclosed in parenthesis, follow the array name. See the Matrix ROM Reference Manual and the following sequence.

```
10  DIM X$(10)5
20  MAT INPUT X$(4)11
```

In Line 20, the array X$ is redimensioned from a 10 element array with 5 bytes per element to a 4 element array with 11 bytes per element. Thus, four readouts are received when Line 20 is executed. When an array is redimensioned, the new dimensions cannot require more bytes of storage than previously required.

2.83 THE $GIO STATEMENT

The syntax and features of the $GIO statement are presented in detail in the General I/O Instruction Set Reference Manual.

The $GIO statement, together with the $UNPACK statement in the General I/O ROM, can be used to develop some of the most time-efficient programming techniques for receiving multi-readouts via a Model 2252A interface and also converting the data into the System 2200 internal numeric format. Furthermore, for applications where each readout represents several discrete values, the $UNPACK statement can be used subsequent to a $GIO or another input operation) to develop an efficient technique for separating discrete information and simultaneously converting the data, if required.

The $GIO statement is unlike any other BASIC language statement. The statement provides, within the framework of the System 2200 BASIC language, a "general input/output" capability designed to meet the individual signal-sequence requirements of a wide variety of I/O peripherals and to optimize I/O performance for these peripherals. Signal sequences for I/O operations can be custom-tailored by specifying a microcommand sequence in the $GIO format. Specification of a microcommand sequence is similar to machine language programming and can be made directly or indirectly in a $GIO statement. A single microcommand prescribes a fundamental operation consisting of several steps; a sequence of microcommands structures a customized operation. Each microcommand is represented by a four hexdigit code (two bytes). The General I/O Instruction Set Reference Manual describes seventeen available categories of microcommands and presents subcategories in a set of tables. However, because of the generality of the $GIO statement, many microcommands cannot be used for Model 2252A applications.

Because the Model 2252A interface controller is designed for input-only operations, and incoming data is limited to a prescribed format by the particular interfaced device in each application, only one microcommand should be needed to customize a $GIO operation for a majority of possible applications. (see Examples 2-20 through 2-23).

Though optional, specification of a device address in a $GIO statement is recommended. (If neither an absolute nor indirect address is specified, the system enables the controller board currently selected for TAPE-class operations.) The $GIO syntax requires a three-hexdigit-address even though the device type hexdigit is not utilized during statement execution. Device type = 2 is used in Examples 2-20 through 2-22 for consistency with all operations except DATALOAD BT which requires device type = 6 for Model 2252A applications.

Example 2-20. Multi-readouts via a $GIO Statement

The following sequence receives 20 readouts via a Model 2252A interface (address 25A) and stores the readouts in the X$-array, defined as a one-dimensional array with a storage capacity for 20 elements with 12 bytes per element. (Each readout consists of a sign, 10 BCD digits, and a carriage-return.)

```
10  DIM X$(20)12, (X20), R$10
20  $GIO /25A (C640, R$) X$()
30  D$ = HEX(000D)
40  $UNPACK (D=D$) X$() TO X()
```

In Line 20, the signal sequence of the $GIO operation is defined by the single microcommand C640. R$ is the specified "arg-2" register (required by the statement syntax); however, the information normally stored in an arg-2 register serves no appropriate function for the microcommand used here. Therefore, the size of the array (total number of bytes) should be a multiple of the number of bytes per readout. The exact number of bytes per readout is equal to the setting on the number-of-digits switches plus two (one for the sign and one for the carriage-return).

In Line 30, D$ is the delimiter specification variable for the $UNPACK operation. Such a variable must contain a minimum of two bytes (any additional bytes are ignored by the system). The first byte defines which one of the alternative unpacking procedures is to be used during statement execution, and the second byte defines the actual delimiter code for the operation; for example, $(0D)_{16}$ represents the carriage-return character which serves as the delimiter for each readout. (See the General I/O Instruction Set Reference Manual for a discussion of the alternative unpacking procedures.)

Example 2-21. Specifying the Number of Readouts Using an Array Modifier

In Example 2-20, 240 bytes are transferred to the CPU during execution of Line 20 since the size of the X$-array is defined to be 20 elements with 12 bytes per element, according to the dimension statement in Line 10. However, the number of bytes to be transferred to the CPU can be specified within a $GIO statement by using an "alpha array modifier" as demonstrated in Line 20 of the following sequence. (The alpha array modifier is defined in the General I/O Instruction Set manual.)

```
10  DIM Y$(40)16, Y(20), R$10
20  $GIO /25A (C640, R$) Y$() <1,240>
30  D$ = HEX(000D)
40  $UNPACK (D=D$) Y$() TO Y()
```

Even though the Y$-array is dimensioned with 40 elements of 16 bytes per element, the alpha array modifier <1,240> implies that the operation in Line 20 is to be performed for 240 bytes beginning with the first byte of Y$(). Since the $UNPACK statement treats the array to be unpacked as a contiguous group of characters, the length of each array element need not match the number of bytes per readout (assumed here to be twelve).

Example 2-22. Specifying a Termination Code for a $GIO Operation

The following sequence demonstrates the use of $GIO and $UNPACK statements to receive and convert multi-readouts until a specified termination code is received; namely, HEX(3C) corresponding to input of a code $(1100)_2$ on input circuits ordinarily allocated for one of the incoming BCD digits (see Table 2-1). For simplicity, the assumption is made in Line 10 that the interfaced device does not supply more than 100 readouts before the non-numeric termination code is likely to occur.

Program Sequence	Comments
`10  DIM X$(100)12, X(100), R$10`	Receive 100 readouts, maximum.
`20  R$ = HEX(3C)`	Store termination code as first byte of R$.
`30  $GIO /25A (C600, R$) X$()`	Receive data until HEX(3C) is received.
`40  N = 256*VAL(STR(R$,9,1))+VAL(STR(R$,10,1))`	Calculate total number of bytes
`50  N = INT(N/12)*12`	received (excluding readout containing HEX(3C) code).
`60  D$ = HEX(030D)`	Define delimiter code $(0D)_{16}$ and unpacking procedure $(03)_{16}$.
`70  $UNPACK (D=D$) X$() <1,N>  TO X()`	Convert received data to numeric values.
`80  R$ = HEX(0D)`	Redefine termination code.
`90  $GIO /25A (C600,R$) F$`	Complete transfer of remaining

In Example 2-22, Line 40 utilizes the fact that the ninth and tenth bytes of the arg-2 register (specified as R$ in Line 30) contain the count (stored as a two-byte binary value) of the number of bytes received during the $GIO operation in Line 30. Note that the count, defined as N, in Lines 40 and 50 is used in the alpha array modifier of Line 70 to limit the number of bytes to be converted.

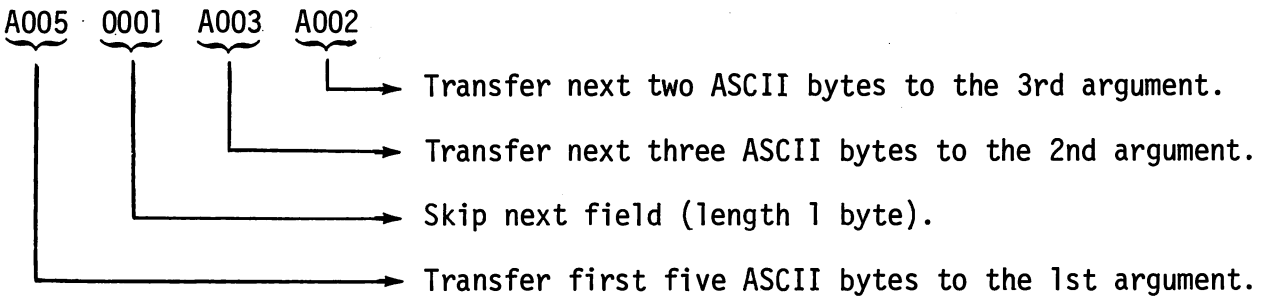
Example 2-23. Separating Readout Data via a $UNPACK Statement

The following sequence demonstrates use of the field format specification in a $UNPACK statement to separate parts of a readout received via a Model 2252A interface. Assuming the number-of-digits setting on the interface is 10, Line 20 stores 12 bytes in **X$(), starting with byte five. Also,** assuming the five high-order bytes (beginning **with** the sign byte) are to be separated and stored in A$, then **bytes 7, 8, 9** are to be stored in B$, and **the 10th and 11th** bytes are to be stored in C$, Line 30 defines the field specification variable for the unpacking operation. (The twelfth byte, the carriage-return, is not included in the unpacking operation.)

```
      Program Sequence                              Comments

10  DIM X$(30)1,R$10
20  $GIO /25A (C640, R$) X$()<5,12>                 Receive and store 12-byte readout.
30  F$ = HEX(A0050001A003A002)                      Define the field format.
40  $UNPACK (F=F$) X$() <5,12> TO A$,B$,C$          Separate the readout by fields.
```

The field format specification variable, F$, is interpreted as follows:



Alternatively, the three fields can be converted to numeric values during the unpacking operation by rewriting Lines 30 and 40 as follows:

```
30  G$ = HEX(10050001100310O2)
40  $UNPACK (F=G$) X$() <5,12> TO A,B,C
```

In the field specification variable G$, the specification 1005 means "convert the first five bytes from an ASCII format to a numeric format". Similarly, 1003 means "convert the next three bytes from an ASCII format to a numeric format". (See the General I/O Instruction Set manual for a discussion of field specification codes.)

APPENDIX A -- DEVICE ADDRESSES FOR SYSTEM 2200 PERIPHERALS

Device Categories & Model Numbers	Standard Addresses*
Keyboards (2215, 2222, 2223)	001, 002, 003, 004
CRTs (2216, 2216A)	005, 006, 007, 008
Cassette Drives (2217, 2218)	10A, 10B, 10C, 10D, 10E, 10F
Line Printers (2221, 2231, 2261, 2272)	215, 216
Output Writer (2201)	211, 212
Plotters (2202, 2212, 2232)	413, 414
Disk Drives (2230, 2240, 2242, 2243, 2260)	310, 320, 330
Mark Sense Card Reader (2214)	517
Hopper Feed Card Readers (2234A, 2244A)	628
Punched Tape Reader (2203)	618
I/O Interface Controller, RS-232-C Compatible (2207A)	019, 01A, 01B Input 01D, 01E, 01F Output
Asynchronous Telecommunications Controller (2227)	219, 21A, 21B Input 21D, 21E, 21F Output
I/O Interface Controller, 8-Bit-Parallel (2250)	23A, 23C, 23E Input 23B, 23D, 23F Output
BCD Input Interface Controllers (2252, 2252A)	25A, 25B, 25C, 25D, 25E, 25F
Digitizers (2262-1, -2, -3)	25A, 25B, 25C, 25D, 25E, 25F

*Since a System 2200 configuration may include more than one device belonging to a particular category, more than one standard address is listed for most categories. If a configuration has only one device in a particular category, that device is set up with the first device listed for the category (e.g., a single cassette drive is assigned the address 10A). If a configuration has more than one device in a particular category, the addresses listed for the category are assigned sequentially (e.g., two cassette drives in one configuration are assigned the addresses 10A and 10B). The controller board into which a device is plugged has a label showing the assigned address. Address uniqueness is essential for the devices in one configuration.

APPENDIX B -- DEVICE SELECTION FOR SYSTEM 2200 OPERATIONS

Five address codes are designated as Primary Device Addresses for the System 2200. Whenever the system is Master Initialized (i.e., power is turned off and then on again), each primary device becomes the default address for one or more of the eight I/O classes into which all System 2200 I/O operations are divided. See Table B-1.

Table B-1. Primary/Default Device Addresses

Address Code	Primary Address for Device Category	Default Address for I/O Classes
001	Keyboards	CI (Console Input) INPUT
005	CRTs	CO (Console Output) PRINT LIST
10A	Cassette Drives	TAPE
310	Disk Drives	DISK
413	Plotters	PLOT

In Chart B-1, most of the operations included in each I/O class are shown. Four of the I/O classes are identified by a parameter which is identical to one of the operations included in the class (see the classes INPUT, PRINT, LIST, and PLOT).

To access a nonprimary device for a particular operation, use a SELECT statement to assign the nonprimary device address to the I/O class parameter which "governs" the operation. For example, the statement

```
SELECT PRINT 215
```

instructs the system to access the device with address 215 (usually a Line Printer) for all subsequent output from execution of Program Mode PRINT and HEXPRINT statements and all output from execution of PRINTUSING statements. (As shown in Chart B-1, output from execution of Immediate Mode PRINT and HEXPRINT statements appears on the device last selected for CO-class operations.)

A single SELECT statement can be used to select two or more devices for subsequent operations belonging to different I/O-classes. For example,

```
SELECT LIST 215, PRINT 211, TAPE 10C
```

When selecting peripheral devices, keep in mind that an I/O-class-parameter and a three-hexdigit-address are required, and a comma must separate the devices when selecting more than one.

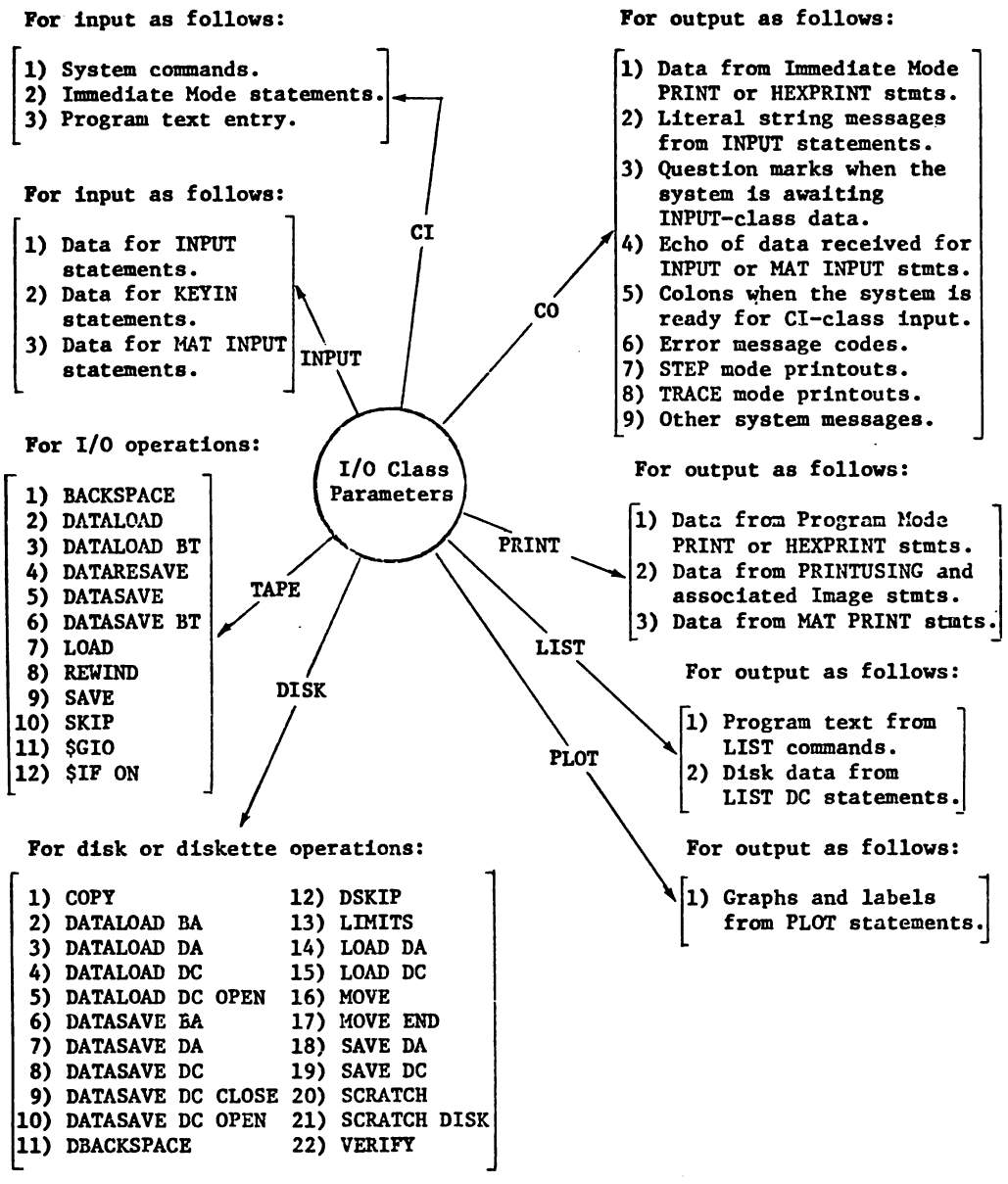
To reselect the CRT for PRINT-class operations, use the statement

```
SELECT PRINT 005
```

or Master Initialize the system if the memory can be cleared at this point.

A SELECT statement "assigns" the specified device address to the specified I/O class parameter. The assignment is analogous to setting an I/O class rotor switch which includes the device-address-options for that class. All subsequent operations belonging to the particular class are "switched" to the designated device until the system encounters another SELECT statement which changes the device address for the I/O class.

Chart B-1. I/O Class Parameters and Operations



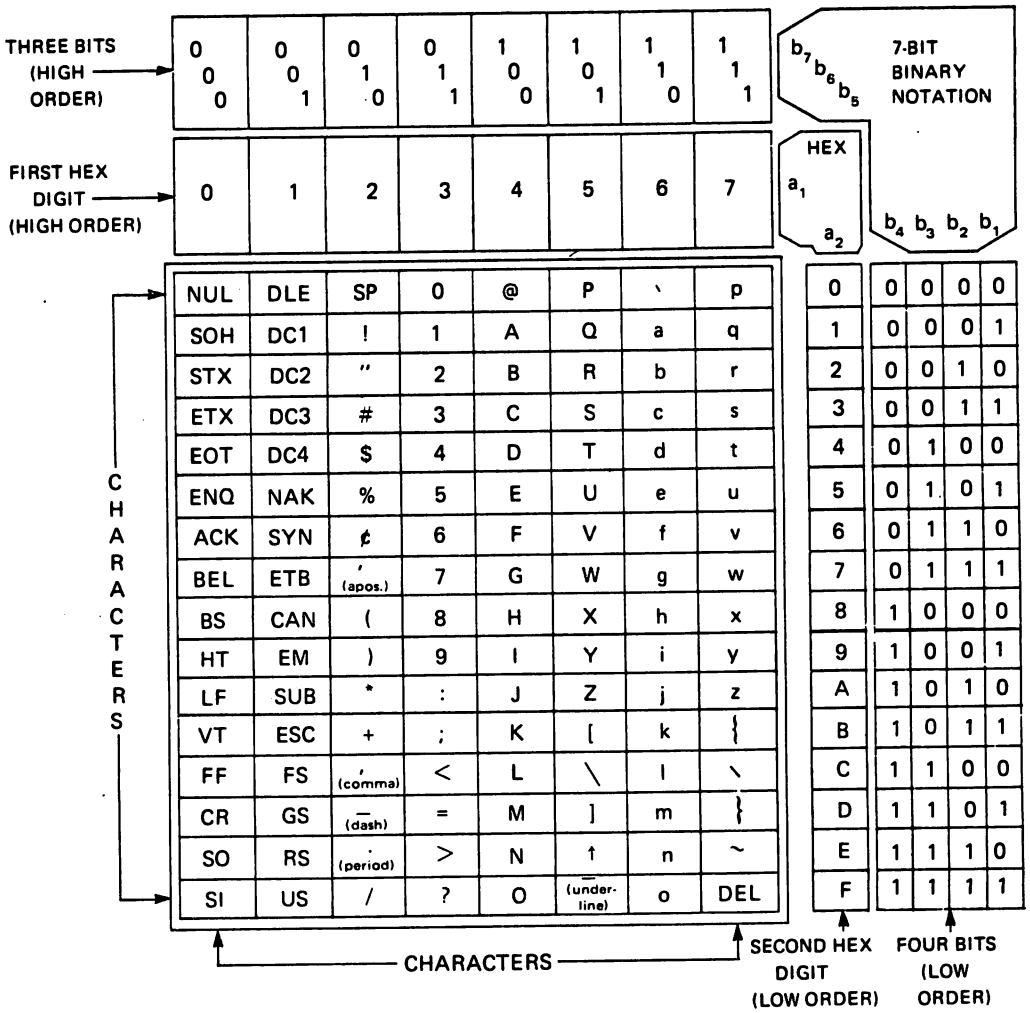
APPENDIX C - ASCII CONTROL AND GRAPHIC CHARACTERS IN HEXADECIMAL AND BINARY NOTATION

FORMATS:

HEXADECIMAL CODES: HEX (a_1 a_2)

7-BIT BINARY CODES: (b_7 b_6 b_5 b_4 b_3 b_2 b_1)

Note:
System 2200 Character Set--
8-bit codes ($b_8b_7b_6b_5b_4b_3b_2b_1$)
b_8=0, b_7 through b_1=ASCII.



	000	001	010	011	100	101	110	111
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	¢	6	F	V	f	v
7	BEL	ETB	' (apos.)	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	, (comma)	<	L	\	l	|
D	CR	GS	– (dash)	=	M	]	m	}
E	SO	RS	. (period)	>	N	↑	n	~
F	SI	US	/	?	O	_ (underline)	o	DEL

LEGEND FOR ASCII CONTROL CHARACTERS

NUL	Null	DLE	Data Link Escape
SOH	Start of Heading	DC1	Device Control 1
STX	Start of Text	DC2	Device Control 2
ETX	End of Text	DC3	Device Control 3
EOT	End of Transmission	DC4	Device Control 4
ENQ	Enquiry	NAK	Negative Acknowledge
ACK	Acknowledge	SYN	Synchronous Idle
BEL	Bell (audible or attention signal)	ETB	End of Transmission Block
BS	Backspace	CAN	Cancel
HT	Horizontal Tabulation (punched card skip)	EM	End of Medium
LF	Line Feed	SUB	Substitute
VT	Vertical Tabulation	ESC	Escape
FF	Form Feed	FS	File Separator
CR	Carriage Return	GS	Group Separator
SO	Shift Out	RS	Record Separator
SI	Shift In	US	Unit Separator
		DEL	Delete

APPENDIX D -- SPECIFICATIONS

Size

Height ..... 7 in. (17.8 cm)
Width ..... 13.5 in. (34.3 cm)
Depth ..... 1.25 in. (3.2 cm)

Weight

3 lb (1.36 kg)

Power Requirements

Supplied by the CPU.

Connector

A 50-pin female Amphenol connector is mounted on the unit.
A 50-pin male Amphenol connector, to be wired to the cable from a device, is supplied with the unit.

Operating Environment

50°F to 90°F (10°C to 32°C)
20% to 80% relative humidity

Switches

Internal: Device-address-switch on printed circuit board.

External: Six logic-level-selection switches to reverse signal level definitions, as required, for the following signals: (1) the input strobe, (2) the sign bit, (3) all BCD or discrete input, (4) the execute signal, (5) the transfer-in-progress signal, and (6) the end-of-transfer output strobe.

Four number-of-digits switches to define the exact number of BCD digits (or 4-bit groups of discrete data) to be processed per readout.

Switch-Reversible Logic Levels

High-level signals: +2.5 to 5 vdc. (High="1" if switch Up, "0" if Down.)
Low-level signals: 0 to +0.4 vdc. (Low="0" if switch Up, "1" if Down.)

Typical Impedance

Input: 4K ohms
Output: 1K ohms

SPECIFICATIONS (Continued)

Input Strobe Pulse Width

2 microseconds (minimum)

Number Code

BCD (8-4-2-1) Code

Number Size

1-to-10 BCD digits and a sign, or up to 41 discrete bits.

Transfer Format

Parallel

Transfer Rate

Up to 100 readings per second using INPUT or MAT INPUT statements.
Up to 800 readings per second using DATALOAD BT statements.
Up to 1000 readings per second using $GIO statements.

Standard Warranty Applies

APPENDIX E -- SETTING THE MODEL 2252A ADDRESS SWITCH

The 8-pole address switch for the Model 2252A interface controller is located in the lower right corner on the chip side of the printed circuit (P.C.) board (when the board is held with the face plate horizontal and above the P.C. board). Eight rocker-type microswitches are enclosed in a rectangular frame and covered by a removable transparent shield. The microswitches are visible through the shield, but the shield must be removed in order to read the labels on the switch frame. A diagram of the switch frame is shown in Figure E-1 with a grid added for reference purposes only.

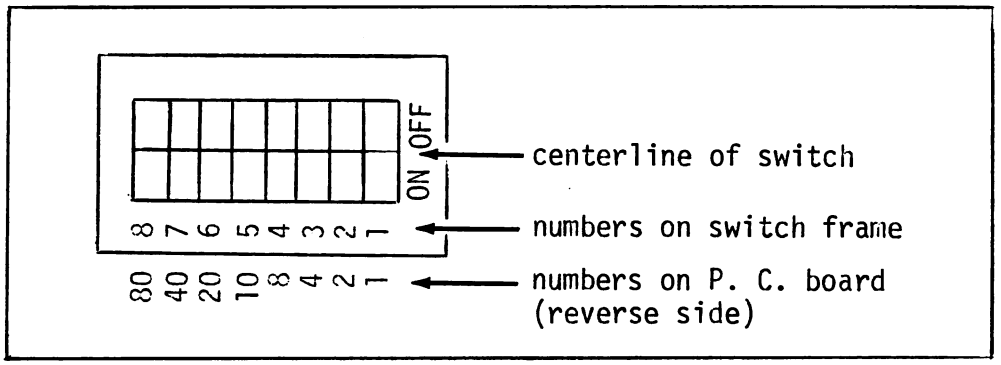


Figure E-1. Frame for the Address Switch

Each of the eight microswitches (not shown in Figure E-1) is identified by a position number printed on the frame. Each rocker-type microswitch pivots about the centerline of the frame. One end of an individual microswitch lies in the OFF-row of the grid; the other end lies in the ON-row of the grid. When one end of a microswitch is DOWN, the other end if UP. The microswitch is turned ON if the end in the ON-row is DOWN.

To avoid confusion when reading the ON-OFF configuration of a switch, look only at the ON-row of the grid. Then translate the DOWN position as ON and the UP position as OFF for each microswitch.

For a System 2200 equipped with one Model 2252A unit, the standard address code is 25A. The standard address code for a second unit in a dual-unit configuration is 25B. These address codes are of the form xyy. The x-digit (the device-type-digit) is not considered when setting an address switch; only the last two hexdigits (i.e., yy) are considered. See Table E-1.

The last two hexdigits of the standard address must be converted into an 8-bit binary number. The leftmost digit in the 8-bit binary number corresponds to the position "8" on the switch frame and the rightmost digit to position "1" on the frame. When a bit in the 8-bit binary number is zero, the microswitch in the corresponding position is turned OFF. When a bit is one, the corresponding microswitch is turned ON (see Table E-1).

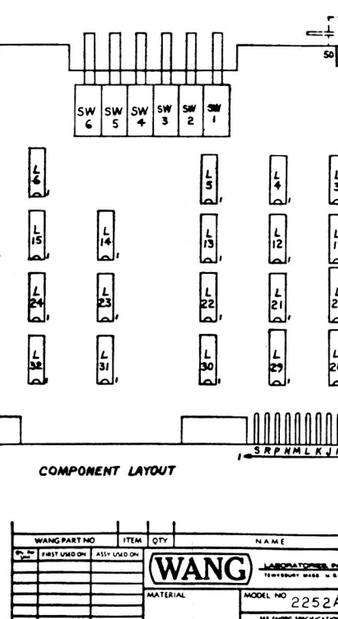
Table E-1. Address Switch Settings for Model 2252A Units

Standard Address	Switch Setting (Hexadecimal)	8-bit Binary Equivalent	Configuration For ON-row of Switch*
25A	5A	01011010	UDUDDUDU
25B	5B	01011011	UDUDDUDD
25C	5C	01011100	UDUDDDUU
25D	5D	01011101	UDUDDDUD
25E	5E	01011110	UDUDDDDU
25F	5F	01011111	UDUDDDDD

*Each microswitch has two visible ends. Look only at the end in the ON-row of the address-switch-frame.
The microswitch is {ON / OFF} if the end in the ON-row is {DOWN (D) / UP (U)}.

LOCATION	W.L. PART NO.	TERM. NO. ±0V	TERM. NO. Vcc+5V
L1-5,9-13,31,32	376-0028	7	14
L6	-0055	7	14
L7,16	-0006	7	14
L8,17	-0080	8	16
L14	-0036	7	14
L15,19,21,22	-0010	7	14
L18,27	-0148	7	14
L20	-0008	8	16
L23,24	-0002	7	14
L25	-0003	7	14
L26,33	-0081	7	14
L28	-0053	8	16
L29	-0087	8	16
L30	-0004	7	14
L34	376-0056	7	14

COMPONENT	W.L. PART NO.
R1-40	
R41-45,59-67	330-3047
R46,48-50,52-54,56-58,68-70	330-3010
R47,51,55	330-4033
C1,4,5	300-1470
C6-13	300-1903
C14,15	300-4022
SW1-6	325-2222M
SW7-10	325-2224
SW11	325-1503
SW11 CAP	325-9047
J1	350-1027
C2,3	300-1480



COMPONENT LAYOUT

ELOC REV 2

WANG PART NO	ITEM	QTY	NAME	MATERIAL	DESCRIPTION
			WANG LABORATORIES, INC. TEWKSBURY, MASS. U.S.A.	BY	DATE / APPROVED BY / DATE
			MODEL NO 2252A	DWN	
			MATERIAL	CHK	E ENGR
			SEE ENGRG SPECIFICATIONS	E C CONTROL	M ENGR / MFG ENGR
			FINISH	TOL EX AS NOTED XX ± .010 FRAC ± 1/64 XXX ± .005 ANG ± 1/30 FINISH	TITLE SCHEMATIC LOGIBLOC 6592 PARALLEL INPUT
				SCALE SHT 1 OF 1	210-6592 / WANG PART NUMBER / E / SIZE / 6592 / DRAWING NUMBER / 2 / REV

INDEX

Preventive Maintenance Information

It is recommended that your equipment be serviced annually. A Maintenance Agreement is available to assure this servicing automatically. If no Maintenance Agreement is acquired, any servicing must be arranged for by the customer. A Maintenance Agreement protects your investment and offers the following benefits:

Preventive Maintenance:

Your equipment is inspected annually for worn parts, lubricated, cleaned and updated with any engineering changes. Preventive maintenance minimizes "downtime" by anticipating repairs before they are necessary.

Fixed Annual Cost:

When you buy a Maintenance Agreement, you issue only one purchase order for service for an entire year and receive one annual billing. More frequent billing can be arranged, if desired.

Further information regarding Maintenance Agreements can be acquired from your local Sales-Service Office.

NOTE

Wang Laboratories, Inc., does not honor Maintenance Agreements for, nor guarantee, any equipment modified by the user. Damage to equipment incurred as a result of such modification is the financial responsibility of the user.

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.

700-3625A

TITLE OF MANUAL: **2252A SCANNING INPUT INTERFACE CONTROLLER**

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.
ONE INDUSTRIAL AVENUE
LOWELL, MASSACHUSETTS 01851

Attention: Technical Writing Department

Fold

Cut along dotted line.

Printed in U.S.A.
13-1019

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

WANG GESELLSCHAFT M.B.H.
Merlingengasse 7
A-1120 Vienna, Austria
TELEPHONE 85.85.33
Telex: 74640 Wang a

WANG EUROPE S.A./N.V.
250, Avenue Louise
1050 Brussels, Belgium
TELEPHONE 02/6400617
Telex: 61186

WANG DO BRASIL COMPUTADORES LTDA.
Rua Barao de Lucena No. 32
Botafogo ZC-01 20,000
Rio de Janeiro RJ, Brasil
TELEPHONE 226-4326, 266-5364
Telex: 2123296 WANG BR

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

WANG FRANCE S.A.R.L.
Tour Gallieni, 1
78/80 Ave. Gallieni
93170 Bagnolet, France
TELEPHONE 33.1.3602211
Telex: 680958F

WANG PACIFIC LTD.
9th Floor, Lap Heng House
47-50 Gloucester Road
Hong Kong
TELEPHONE 5-274641
Telex: 74879 WANG HX

WANG COMPUTER LTD.
Shindaiso Building No. 5
2-10-7 Dogenzaka Shibuya-Ku
Tokyo 150, Japan
TELEPHONE (03) 464-0644
Telex: 2424909WCLTKO J

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

WANG COMPUTER LTD.
302 Great North Road
Grey Lynn, Auckland
New Zealand
TELEPHONE Auckland 762-219
Telex: CAPENG 2826

WANG DE PANAMA (CPEC) S.A.
Apartado 6425
Calle 45E, No. 9N. Bella Vista
Panama 5, Panama
TELEPHONE 69-0855, 69-0857
Telex: 3282243

WANG COMPUTER PTE., LTD.
Suite 1801-1808, 18th Floor
Tunas Building, 114 Anson Road
Singapore 2, Republic of Singapore
TELEPHONE 2218044, 45, 46
Telex: RS 24160 WANGSIN

WANG COMPUTERS (SO. AFRICA) PTY. LTD.
Corner of Allen Rd. & Garden St.
Bordeaux, Transvaal
Republic of South Africa
TELEPHONE (011) 48-6123
Telex: 960-86297

WANG SKANDINAVISKA AB
Pyramidvaegen 9A
S-171 36 Solna, Sweden
TELEPHONE 08/27 27 98
Telex: 11498

WANG S.A./A.G.
Markusstrasse 20
CH-8042 Zurich 6, Switzerland
TELEPHONE 41-1-60 50 20
Telex: 59151

WANG INDUSTRIAL CO., LTD.
7 Tun Hwa South Road
Sun Start Tun Hwa Bldg.
Taipei, Taiwan
Republic of China
TELEPHONE 7522068, 7814181-3
Telex: 21713

WANG ELECTRONICS LTD.
Argyle House, 3rd Floor
Joel Street
Northwood Hills
Middlesex, HA6INS
TELEPHONE Northwood 28211
Telex: 923498

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

DATA CENTER DIVISION
20 South Avenue
Burlington, Massachusetts 01803
TELEPHONE (617) 272-8550

WANG COMPUTER SERVICES
One Industrial Avenue
Lowell, Massachusetts 01851
TELEPHONE (617) 851-4111
TWX 710-343-6769
Telex: 94-7421

WANG INTERNATIONAL TRADE, INC.
One Industrial Avenue
Lowell, Massachusetts 01851
TELEPHONE (617) 851-4111
TWX 710-343-6769
Telex: 94-7421

WANG LABORATORIES, INC.

ONE INDUSTRIAL AVENUE, LOWELL, MASSACHUSETTS 01851, TEL. (617) 851-4111, TWX 710 343-6769, TELEX 94-7421

Printed in U.S.A.
700-3625A
4-77-1M
Price: see current list