WANG

2200

2207A I/O Interface Controller User Manual

2200
2207A I/O Interface Controller User Manual

Copyright © Wang Laboratories, Inc., 1976
700-3364

WANG

WANG LABORATORIES, INC., ONE INDUSTRIAL AVENUE, LOWELL, MA 01851 • TEL: 617/459-5000, TWX 710-343-6769, TELEX 94-7421

PREFACE

Information regarding the installation and operation of the Model 2207A I/O Interface Controller is provided in this manual.

Chapter 1 describes the features of the controller and includes some sections of primary interest to persons responsible for interfacing a non-Wang device to a System 2200.

Chapter 2 describes programming techniques for the Model 2207A controller.

CONTENTS

Page

CHAPTER 1 MODEL 2207A FEATURES

1.0 General Information. 1
1.1 Installation 2
1.2 Connector Pin Assignments. 2
1.3 Device Address Codes 3
1.4 Transmission Timing. 4

CHAPTER 2 PROGRAMMING TECHNIQUES

2.0 Introduction 6
2.1 Device Selection 6
2.2 Choosing BASIC Language Operations 8
2.3 Customizing Operations 10
2.4 Controlling a Teletype Terminal. 11

APPENDIX A STANDARD ADDRESSES FOR PERIPHERALS AND CONTROLLERS . . . 13
APPENDIX B I/O-CLASS PARAMETERS AND PRIMARY ADDRESSES 15
APPENDIX C BASIC LANGUAGE SYNTAX AND TERMS. 18
APPENDIX D FUNDAMENTAL I/O STATEMENTS 20

INPUT 21
KEYIN 27
MAT INPUT 29
MAT PRINT 38
PRINT 30
PRINTUSING. 35

APPENDIX E SPECIAL PURPOSE I/O STATEMENTS 39

DATALOAD. 40
DATALOAD BT 43
DATASAVE. 46
DATASAVE BT 49
LOAD. 50
SAVE. 53

APPENDIX F CUSTOMIZED I/O STATEMENTS. 55

$GIO. 56

APPENDIX G ASCII CONTROL AND GRAPHIC CHARACTERS 60
APPENDIX H SETTING ADDRESS SWITCHES ON THE MODEL 2207A CONTROLLER . 61
APPENDIX I MODEL 2207A SPECIFICATIONS 63

EQUIPMENT MAINTENANCE 64

CUSTOMER COMMENT FORM . . . . Last Page

TABLES

Page

Table 1-1. Transmission Timing for the Model 2207A Controller. . . . 5
Table 2-1. Summary of Input Operations 9
Table 2-2. Summary of Output Operations. 10
Table A-1. Device Type Codes for Model 2207A I/O Operations. 14
Table B-1. Primary Addresses for the System 2200 16
Table H-1. Standard Address Switch Settings
for Model 2207A Controllers 62

CHARTS

Chart B-1. I/O Class Parameters and Operations 17

FIGURES

Figure 1-1. Model 2207A I/O Interface Controller. 1
Figure 1-2. Model 2207A Connector Pin Assignments 2

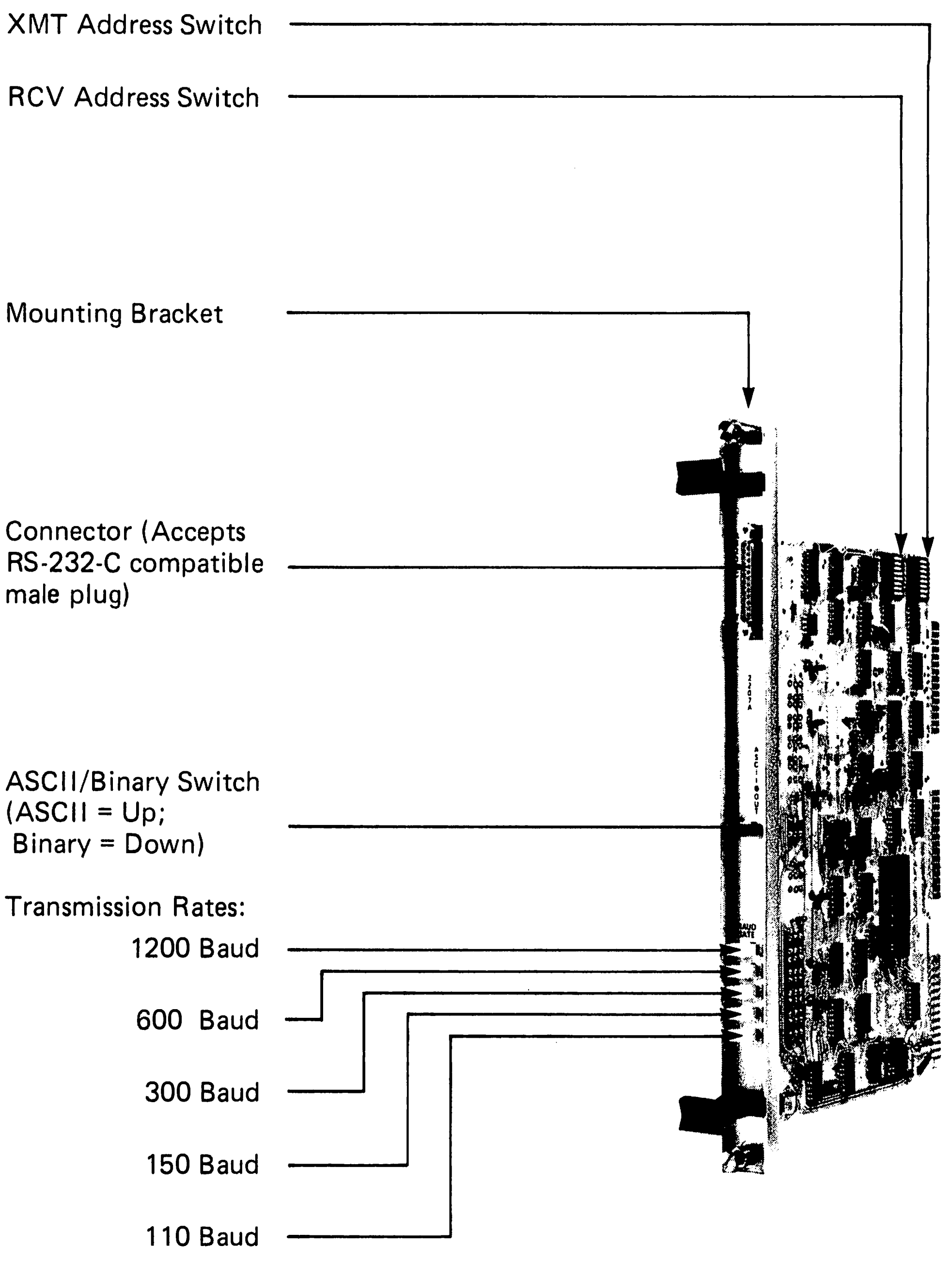


Figure 1-1. Model 2207A I/O Interface Controller

CHAPTER 1

MODEL 2207A FEATURES

1.0 GENERAL INFORMATION

The Wang Model 2207A I/O Interface Controller plugs into any I/O slot in a System 2200 Central Processing Unit (CPU). A 25-pin connector on the controller facilitates direct hookup of an RS-232-C compatible non-Wang device. Examples include devices such as the following:

. an RS-232-C compatible Teletype®,

. a Teletype-equivalent terminal, or

. an RS-232-C compatible, asynchronous-transmission laboratory instrument.

The device to be interfaced to a System 2200 should be equipped with a suitable cable (maximum length 50 feet, i.e., 15.2 meters) and a compatible male plug.

The Model 2207A connector pin assignments are described in Section 1.2 for anyone planning to interface a device to a System 2200 CPU. The EIA (Electronic Industries Association) Standard RS-232-C provides recommendations for interchange circuits and connector pin assignments for equipment designed to qualify as RS-232-C compatible; however, generally speaking, users of an operational System 2200/2207A do not need to read EIA Standard RS-232-C.

The Model 2207A controller is designed to support asynchronous transmission line speeds up to 1200 baud. The controller has five "baud rate" switches labeled 110, 150, 300, 600 and 1200. See Figure 1-1. A desired transmission rate is selected by pushing the corresponding switch in. Table 1-1 in Section 1.4 converts the rate in baud to characters per second and milliseconds per character.

The controller supports two operational modes, called ASCII and Binary, designed to handle different data formats. The ASCII/Binary switch (shown in Figure 1-1) determines the operational mode as follows: ASCII = switch out; Binary = switch in.

In the ASCII mode, the controller treats the data format as 7-level ASCII code plus an even-parity high-order bit. For input operations, the parity bit is ignored, i.e., the high-order eighth bit is interpreted as "0" always. For output operations, the high-order eighth bit is transmitted as a "0" or "1" depending upon the value needed to conform to an even-parity condition (i.e., to make the sum of the 1-bits, including the parity bit, an even number).

Furthermore, in the ASCII mode, the controller automatically decodes a standard Teletype BREAK signal and sends a HALT/STEP signal to the CPU, and decodes a standard ESC (Escape) signal and sends a RESET signal to the CPU. Therefore, if desired, a Teletype can be installed as a console input device in lieu of a Wang keyboard. See Section 1.3.

In the Binary mode, the controller treats the data format as 8-bits per character for input and output operations. Decoding of BREAK and ESC signals is inhibited during input operations.

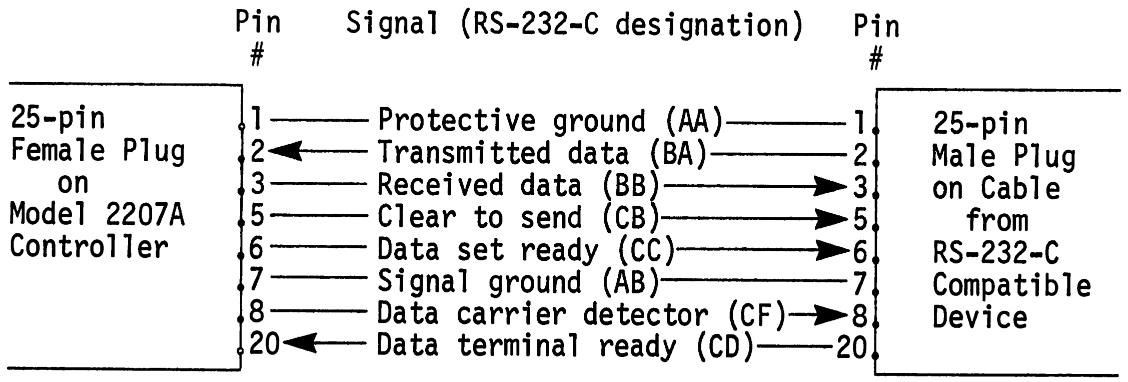
1.1 INSTALLATION

Installation of a Model 2207A controller is the responsibility of a Wang Service Representative who should be called when the controller arrives.

1.2 CONNECTOR PIN ASSIGNMENTS

The information in this section can be ignored by readers not responsible for interfacing a device to a System 2200.

Special attention should be given to the pin assignments for the Model 2207A connector, shown in Figure 1-2. The female plug on the controller accepts an RS-232-C compatible male plug directly; <u>no modem is required</u>. Therefore, from the controller viewpoint, data is received on Pin 2 and transmitted on Pin 3. On the other hand, from the viewpoint of an interfaced device (as shown in Figure 1-2), data is transmitted via Pin 2 and received via Pin 3.


Pin #
Signal (RS-232-C designation)
Pin #
25-pin Female Plug on Model 2207A Controller
1 — Protective ground (AA) — 1
2 ← Transmitted data (BA) — 2
3 — Received data (BB) → 3
5 — Clear to send (CB) → 5
6 — Data set ready (CC) → 6
7 — Signal ground (AB) — 7
8 — Data carrier detector (CF) → 8
20 ← Data terminal ready (CD) — 20
25-pin Male Plug on Cable from RS-232-C Compatible Device

NOTE: The interfaced device must use Pins 1, 2, 3, 7 and 20. Pins 5, 6 and 8 in the Model 2207A connector are tied to the positive level voltage defined by Standard RS-232-C. The interfaced device must supply a positive level voltage on Pin 20.

Figure 1-2. Model 2207A Connector Pin Assignments (From Viewpoint of Interfaced Device)

1.3 DEVICE ADDRESS CODES

To control 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 or one channel of a dual-channel controller.

When a particular I/O statement is executed, the yy-digits in the device address determine which controller in the CPU chassis is enabled for the operation, and the x-digit determines which microcode routines are used by the system to implement the operation.

Each controller has at least one 8-pole address switch. A dual-channel controller such as the Model 2207A has two 8-pole address switches. An 8-pole switch has eight tiny rocker-type switches whose on-off configuration represents an 8-bit binary number corresponding to the yy-digits in a three-hexdigit address. See Figure 1-1.

Usually, an address switch is set by Wang Laboratories before a controller is shipped from the factory or by a Wang Service Representative when a controller is installed in a CPU. Since an address switch is neither visible nor accessible after a controller is mounted in a CPU chassis, a label is attached to the faceplate of each controller to show its address or addresses.

On Model 2207A controllers sent to customers who assume responsibility for installation of the controller, the address switches may not be set prior to shipment. Therefore, information regarding address switches is provided in Appendix H.

When a Model 2207A controller (or any controller) is installed in a CPU, care must be exercised to ensure address uniqueness with respect to other controller boards mounted in the same chassis. For this reason, Wang Laboratories has adopted a set of standard address codes to be used unless a customer's requirements indicate a different code is desirable.

In every System 2200 configuration, one controller (usually a keyboard controller) must be set up with the address 001 for console input operations, i.e., for entry of system commands, program text, and immediate mode statements. Another controller (usually a CRT controller) must be set up with the address 005 for console output operations which provide many automatic system messages to prompt programmers and operators.

NOTE:

If a Model 2207A controller is used to interface a Teletype to a System 2200 in lieu of a keyboard and CRT, the address switches on the controller must conform to settings ordinarily used for the console keyboard and CRT; otherwise, the standard Model 2207A settings are appropriate whether a Teletype or some other device is being interfaced to the system. Anyone installing a Model 2207A controller should see Appendix A for a list of standard addresses.

If more than one Model 2207A controller is installed in a CPU chassis, Appendix A gives the standard addresses for additional units.

Chapter 2 contains information regarding the use of device address codes for program control of I/O operations.

1.4 TRANSMISSION TIMING

The Model 2207A controller transmits and receives data asynchronously, i.e., character-by-character with each character initialized by a start element and terminated by a stop element. In asynchronous transmission, the start element (usually a logic "0" voltage level) is equivalent in duration to one data-bit interval, but the stop element (usually a logic "1" voltage level) is a variable-length interval since the signal is maintained until the next character is transmitted. The stop element has no maximum duration; however, it does have a minimum duration determined by the design of the equipment. For the Model 2207A controller, the minimum length stop element is equivalent to 2.0 data-bit intervals. The minimum length stop element occurs when data is being transmitted automatically from equipment such as a Teletype tape unit or some other pre-recorded medium. The variable length stop element occurs when data is transmitted from a keyboard since the time between keystrokes varies.

The total number of bits per character for asynchronous data transmission via the Model 2207A is eleven, effectively, for both ASCII and Binary mode operations. In ASCII mode operations, the character transmission consists of one start element (equal to one data-bit interval), seven data bits, one parity bit, and a stop element equivalent to two data-bit intervals. In Binary mode operations, the character transmission consists of one start element, eight data bits, and a stop element equivalent to two data-bit intervals. Therefore, in either mode, the number of data bit intervals per character is eleven.

In Table 1-1, the transmission timing for the Model 2207A controller is converted to characters per second and milliseconds per character for the five switch-selectable baud rates. The values in the table represent instantaneous line transmission rates, not data throughput rates. In any configuration, data throughput is limited by such factors as the transfer rate of I/O peripherals and the overhead time required by the program controlling data transfer.

Only one baud rate button can be pushed in; the button corresponding to a previously selected rate automatically springs out when another rate is selected. Choice of an appropriate rate depends upon the specifications of the device plugged into a Model 2207A controller.

Table 1-1. Transmission Timing for the Model 2207A Controller

Baud Rate	Characters per Second	Milliseconds per Character
110	10.0	110.0
150	13.6	73.3
300	27.3	36.7
600	54.5	18.3
1200	109.1	9.2

CHAPTER 2

PROGRAMMING TECHNIQUES

2.0 INTRODUCTION

Some programming techniques for control of equipment interfaced to a System 2200 CPU via a Model 2207A I/O Interface Controller are presented in this chapter. For the information to be completely meaningful, a reader must be familiar with the System 2200 BASIC language and general programming techniques. Anyone unfamiliar with the system should read the BASIC Programming Manual provided with the CPU.

The general forms of the I/O statements discussed in this chapter are presented in Appendices D, E, and F.

2.1 DEVICE SELECTION

Since a System 2200 can be configured with several input and output devices, it is necessary to have a method of identifying the particular device to which, or from which, data is to be transferred. For some System 2200 I/O operations, up to three methods of device identification are possible.

A unique address (or, in some cases, a pair of addresses) is associated with each I/O device in a configuration whether the device is a Wang peripheral, an interfaced non-Wang device, or part of the console I/O equipment. Each device address is a three-hexdigit (12-bit) number of the form xyy, where x represents a _device type code_ and yy represents the _preset address_ of the controller into which a particular device is plugged.

When executing an I/O operation, the system utilizes the x-digit (the 4 high-order bits) in the address to determine which microcode routines are to be used to implement the operation, and the yy-digits (the 8 low-order bits) to determine which controller is to be accessed for the operation.

The uniqueness of a device address is established by its yy-digits which correspond to a binary value set on an 8-pole address switch. The value must be unique with respect to the values set on address switches for all the other controllers in the same CPU.

Since address switches are set before controllers are mounted in a CPU chassis and are not visible thereafter, each controller has a label showing its address or addresses. Labels usually specify three hexdigits by including a device type code as well as the preset controller address -- because the BASIC language syntax always requires three-hexdigit addresses for I/O operations.

Often, one device type code is appropriate for all the I/O operations supported by a particular controller, e.g., x=1 for cassette drive controllers, x=3 for disk controllers. In some cases, several different device type codes are appropriate for the operations supported by a controller, e.g., x=0, 2 or 4 can be used with printer controllers to produce different results. See the PRINT statement in Appendix D.

Different device type codes are used with the controller described in this manual. Appropriate codes are indicated in the discussions of particular applications.

Appendix A gives standard addresses for device categories such as keyboards, CRT's, printers, plotters, and interface controllers. Valid device type codes are described with respect to several I/O operations and device categories.

Once an appropriate device type code and the preset controller address are known, there are alternative ways to specify the device address for an I/O operation. Three methods, with illustrations, follow.

Method 1. Specify the address directly in the I/O statement, using a slash (if permitted by the syntax), e.g.,

```
100 DATALOAD BT /61D, X$()
```

Method 2. Specify the address indirectly in the I/O statement, using a file number (if permitted by the syntax and if the address has been assigned to the file number by a SELECT statement), e.g.,

```
100 SELECT #3 61D
110 DATALOAD BT #3, X$()
```

Method 3. Use a SELECT statement to select the device address for the I/O-class parameter associated with the operation; then, no address need be specified for subsequent operations belonging to that class, e.g.,

```
100 SELECT TAPE 61D
110 DATALOAD BT X$()
```

Method 3 can be used for every I/O operation and must be used for any operation which does not permit specification of a direct or indirect address in the BASIC statement. Methods 1 and 2, even when permitted, are always optional.

NOTE:

When an optional address is omitted in an I/O statement, the default address is the address last selected for the I/O-class parameter associated with the operation; however, if no address has been selected for the parameter, the system uses the "primary" address -- the address automatically selected for the parameter when the system is Master Initialized. See Appendix B, where the eight I/O-class parameters, the operations associated with each parameter, and the primary address for each parameter are summarized.

2.2 CHOOSING BASIC LANGUAGE OPERATIONS

Generally speaking, several BASIC language I/O statements are feasible for program control of devices interfaced to System 2200 CPU's via Model 2207A controllers. Input statements with built-in signal sequences include INPUT, KEYIN, MAT INPUT, LOAD, DATALOAD, and DATALOAD BT. Output statements with built-in signal sequences include PRINT, PRINTUSING, MAT PRINT, SAVE, DATASAVE, and DATASAVE BT. As indicated in the Preface, some statements discussed in this manual are standard in particular CPU models; other statements are available only if options are added to a standard CPU model.

The characteristics of an interfaced device as well as the requirements of an online application must be known before suitable BASIC language statements can be chosen for programming an application. Since the number of RS-232-C devices which can be interfaced to Wang systems is large and their characteristics diverse, a sample device and application of general interest to most readers is difficult to formulate. Because a Teletype represents a four-in-one device, programming techniques for controlling a Teletype interfaced to a Wang CPU are presented in Section 2.4. Some of the techniques may prove helpful for other applications.

The special features of Wang's built-in I/O operations are described in detail in Appendices D and E. Brief summaries of the statements are given in Tables 2-1 and 2-2 to simplify the task of choosing BASIC language statements for program control of an interfaced device.

Table 2-1. Summary of Input Operations

BASIC Statement	Device Type Code	General Features
INPUT	0 or 2	For control of input devices not requiring initialization signals. Can receive data for storage in several numeric and/or alphanumeric variables, including STR functions and specific array elements. Incoming data stream must be suitably structured with commas and carriage-return characters serving as data separators. See Appendix D.
KEYIN	0 or 2	For device-scanning of input devices or single character input "handshaking" applications. Using a loop, multicharacter input can be received one character at a time from devices with relatively slow data transmission rates. See Appendix D.
MAT INPUT	0 or 2	Receives data for storage in one or more numeric or alphanumeric arrays; otherwise, similar to INPUT statement. See Appendix D.
DATALOAD	4 or 6	Can be used to read data values from formatted punched tapes. Device type 4 automatically sends X-ON code to start a tape reader and an X-OFF to stop the unit. Device type 6 sends an initialization code for forward/reverse reading. May be useful for control of other input devices. See Appendix E.
DATALOAD BT	4 or 6	Can be used to read program text or data from unformatted punched tapes. Transmits X-ON and X-OFF codes automatically if device type 4; transmits initialization code for forward/reverse reading if device type 6. May be useful for control of other input devices. See Appendix E. (Note: With ASCII/Binary switch in Binary, any 8-bit code can be read.)
LOAD	4 or 6	Reads program text from formatted punched tapes. See Appendix E.
$GIO	0	Can be used to customize an input (or output) operation. Particularly useful for devices not suitable for the built-in signal sequences of other statements. See Appendix F.

Table 2-2. Summary of Output Operations

BASIC Statement	Device Type Code	General Features
PRINT	0, 2 or 4	Can be used to output values currently stored in numeric and/or alphanumeric variables, or to output any 8-bit code using a HEX function. Also can be used to output null characters sometimes used as "fill characters". Choice of device type code determines whether a line-feed or null character follows each system generated carriage-return character and whether a line character count is maintained or suppressed. See Appendix D.
PRINTUSING	0, 2 or 4	Outputs values interspersed with text (if desired) according to specifications given in a referenced image (%) statement. See Appendix D.
MAT PRINT	0, 2 or 4	Similar to PRINT except the argument list can specify numeric and/or alphanumeric arrays. See Appendix D.
DATASAVE	4	Outputs data with control information (CR LF RUBOUT RUBOUT) separating each value from the preceding one. Numeric values are output in fixed or floating point formats depending upon the magnitude; alphanumeric values are output without trailing spaces. See Appendix E.
DATASAVE BT	4	Outputs data without control information between values. See Appendix E. (Note: With ASCII/Binary switch in Binary, any 8-bit code can be output.)
SAVE	4	Outputs program text with control information. See Appendix E.
$GIO	0	Can be used to customize an output (or input) operation. Particularly useful for devices not suitable for the built-in signal sequences of other statements. See Appendix F.

2.3 CUSTOMIZING OPERATIONS

If one of Wang's BASIC language I/O statements with a built-in signal sequence is not ideally suited to the characteristics of an interfaced device and its online application, the $GIO statement provides the capability to customize input and output operations using a procedure similar to machine language programming.

Wang's General Input/Output statement fits into the framework of the high-level BASIC language, yet approaches machine language programming because a user can specify a desired signal sequence for an input or an output operation by inserting one or more microcommands in the statement. Each microcommand is represented by a four-hexdigit-code. Since each code represents one or more fundamental operations, the microcommands serve as building blocks which can be assembled in a variety of ways.

The $GIO statement and the microcommands available for programming customized I/O operations are described in the General I/O Instruction Set Reference Manual provided with any central processor which has the $GIO statement in its language set. For completeness, some of the features of the statement are summarized in Appendix F of this manual.

2.4 CONTROLLING A TELETYPE TERMINAL

A Teletype (or similar) terminal usually has four components: a keyboard, a printer, a paper tape punch unit, and a paper tape reader unit. Therefore, when a Teletype terminal is interfaced to a Wang central processor via a Model 2207A controller, the terminal should be viewed as a four-in-one package of peripherals which are accessed via one dual-channel controller board. With such a viewpoint, techniques needed to control a Teletype can be developed by readers familiar with the keyboard and printer programming methods described in the BASIC Language Programming Manual provided with each central processing unit.

Remember one important fact when reading the programming manual -- to use a System 2200 CPU, at least two devices are required: a console input device (usually a Wang keyboard) and a console output device (usually a Wang CRT). If desired, the keyboard and printer components of a Teletype terminal can serve as console I/O devices if a Teletype is plugged into a Model 2207A controller whose input and output channels are assigned the primary addresses usually reserved for a Wang keyboard and CRT (i.e., 001 and 005).

If a Teletype is installed as a console I/O unit, any BASIC language operations included in the I/O classes CI, INPUT, CO, PRINT, and LIST (see Appendix B) are automatically implemented via the Teletype keyboard and printer. However, any operations in the I/O-class TAPE (e.g., DATASAVE, DATALOAD, SAVE) used for control of the Teletype read/punch units are not implemented via those units unless the address 405 is specified in a particular statement, or selected for TAPE operations using a SELECT statement.

On the other hand, if a Teletype serves as a peripheral device in a System 2200 configuration having a Wang keyboard and CRT as the console input and output devices, the "input channel" address (usually 019) must be selected for the I/O-class INPUT in order to enter data via the Teletype keyboard in response to an INPUT statement. Similarly, the Teletype printer is not accessed for operations in the I/O-classes CO, PRINT and LIST unless an "output channel" address (e.g., 01D, 21D or 41D) is selected for the appropriate class. Furthermore, the reader/punch units are not accessed for operations in the I/O-class TAPE unless the "output channel" address (usually 41D) is selected for TAPE-class operations or the address is specified in a particular statement.

Another important fact should be kept in mind when a Teletype printer is installed (or selected) for console output operations -- there is no printable symbol on the Teletype corresponding to the cursor which appears on the CRT to show an operator where the next character will appear. Therefore, on a Teletype printer a "ready condition" is denoted by a colon only, rather than the colon and cursor which appear on the CRT whenever the system is awaiting a command or text entry. Other differences between CRT output and Teletype output can be expected; therefore, notes should be kept by anyone who reads the programming manual and duplicates its console I/O examples on a Teletype.

For most Teletype operations the ASCII/Binary switch on the controller should be in the ASCII position. Also, the 110 baud rate switch should be depressed unless the specifications for the equipment recommend another rate.

APPENDIX A

STANDARD ADDRESSES FOR PERIPHERALS AND CONTROLLERS

Device Category (Model Numbers)	Device Addresses*
Keyboards (2215, 2222, 2223)	001, 002, 003, 004
CRT's (2216, 2216A)	005, 006, 007, 008
Cassette Drives (2217, 2218)	10A, 10B, 10C, 10D, 10E, 10F
Line Printers (2221W, 2231, 2261)	215, 216
Output Writers (2201)	211, 212
Plotters (2202, 2212, 2232A)	413, 414
Disk Drives (2230-1, -2, -3; 2260; 2270-1, -2, -3)	310, 320, 330
Nine-Track Tape Drives (2209)	07B, 07D, 01F
Mark Sense Card Readers (2214)	517
Hopper-Feed Mark Sense/Punch Card Readers (2234A)	628
Hopper-Feed Mark Sense/Punch Card Readers (2244A)	628 618
Punched Tape Readers (2203)	25A, 25B, 25C, 25D, 25E, 25F
I/O Interface Controller, RS-232-C (2207A)	x19, x1A, x1B (Input) x1D, x1E, x1F (Output)
Asynchronous Telecommunications Controller (2227)	x19, x1A, x1B (Input) x1D, x1E, x1F (Output)
I/O Interface Controller, 8-bit-Parallel (2250)	x3A, x3C, x3E (Input) x3B, x3D, x3F (Output)
Scanning Input Interface Controller, BCD 1-to-10 Digit Parallel (2252A)	x5A, x5B, x5C, x5D, x5E, x5F
Communications Controller (2228)	x19, x1A, x1B (Input) x1D, x1E, x1F (Output)

*In some cases, more than one address is listed for a device category. If a System 2200 configuration has only one unit belonging to a particular category (e.g., a Model 2252A controller), the unit is set up with the first address (x5A). If there are two units, one is set up with the address x5A and the other with address x5B. The address is written on a label affixed to each controller. Often a "typical" device-type code is included on the label (e.g., 25A); however, particular I/O operations may require a different device-type code. See Table A-1.

Table A-1. Device Type Codes for Model 2207A I/O Operations

Operations	Device Type Codes	Remarks
INPUT, KEYIN, MAT INPUT	0 or 2	Either code can be used without affecting the procedure. Operations may be used with many peripherals and interface controllers.
PRINT, PRINTUSING, MAT PRINT, HEXPRINT	0, 2, or 4	These operations may be used with many peripherals and interface controllers. Code 0 supplies a line-feed character after every system-generated carriage-return character. Code 2 supplies a null character after every system-generated carriage-return character. Code 4 suppresses the line character count and thereby suppresses the carriage-return normally generated when the count equals the currently selected line-length; otherwise, supplies a line-feed after every system-generated carriage-return.
DATALOAD, DATALOAD BT, LOAD	4 or 6	The procedure is dependent upon the device type code in the address. Use of particular codes is restricted to particular peripherals and controllers.
DATASAVE, DATASAVE BT, SAVE	4	The procedure is dependent upon the device type code in the address. Use of particular codes is restricted to particular peripherals and controllers.
$GIO	0	The device type code is ignored by the system during an I/O operation, but is required by the BASIC language syntax.

APPENDIX B
I/O CLASS PARAMETERS AND PRIMARY ADDRESSES

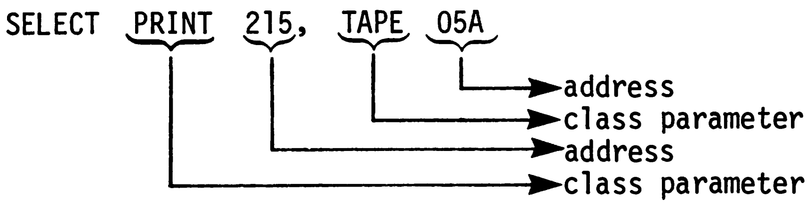
The input/output operations for the System 2200 are divided into eight groups (classes) identified by the following I/O-class parameters: CI (console input), CO (console output), INPUT, PRINT, LIST, TAPE, DISK, and PLOT. The significance of the class parameters is explained in this appendix, and the particular operations belonging to each class are identified in Chart B-1.

The general form for each I/O operation indicates whether address specification in an actual statement is optional or not permitted. However, address specification for an operation is always possible by the following technique. First, determine which I/O-class parameter is associated with a particular operation (e.g., the parameter TAPE is associated with the operation DATALOAD BT). Then, use a statement of the following form to "select" the desired address for the operation:

```
SELECT class-parameter xyy [,class-parameter xyy]...
```

For example,

```
SELECT PRINT 215, TAPE 05A
```



Address selection is unnecessary for five addresses called "primary addresses" for the System 2200. Whenever the system is Master Initialized, a primary address is selected automatically for each of the class parameters, as shown in Table B-1. Thereafter, the primary address associated with a particular parameter (e.g., 005 for PRINT) is used by the system for all subsequent operations belonging to the I/O-class represented by the parameter unless one of the following conditions is in effect:

- a different address is specified directly or indirectly in a particular statement, or
- a SELECT statement is used to select a nonprimary address for the parameter.

An address specified in an I/O statement always has priority over the address currently selected for the class parameter associated with a particular operation.

Specification of a line length (maximum value 255 characters) is optional for three class parameters (CO, PRINT, and LIST), using a SELECT statement of the following form:

```
SELECT class-parameter xyy [(length)] [,class-parameter xyy [(length)]]...
```

For example,

```
    SELECT PRINT 215(80)
```

If no length is specified, the default value is 64. However, when another address is selected for a particular class-parameter without specifying a line length, the last length selected for the parameter applies to the new address.

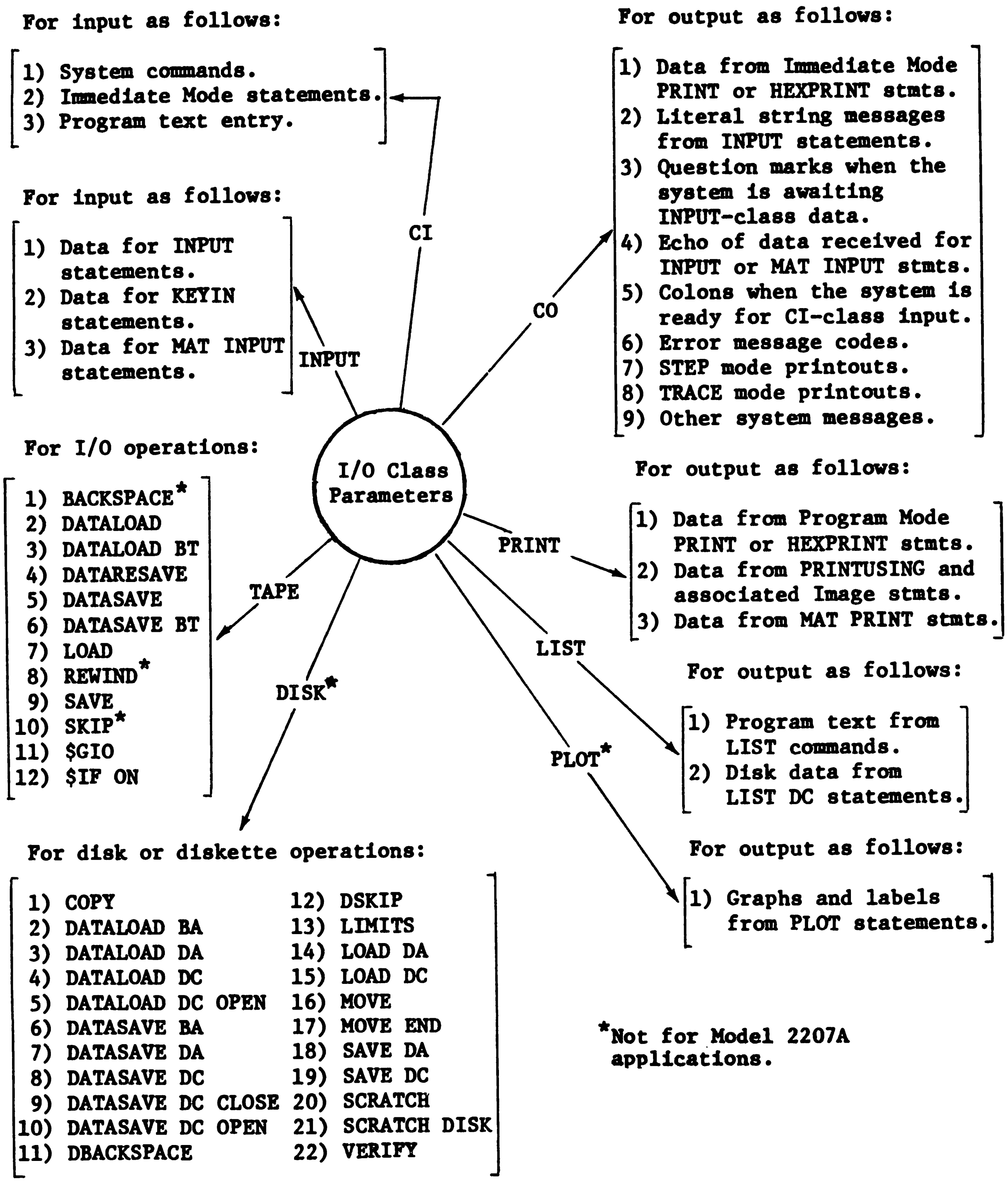
If a CLEAR command is executed, the address last selected for the CI parameter is selected automatically for the parameter INPUT, and the address last selected for the CO parameter is selected for the parameters PRINT and LIST; other parameters are not affected by a CLEAR command.

Table B-1. Primary Addresses for the System 2200
(Addresses Automatically Selected After Master Initialization)

Primary Address*	I/O-class Parameter
001	CI (console input) INPUT
005	CO (console output) PRINT LIST
10A	TAPE
310	DISK
413	PLOT

*Usually, the address 001 accesses a keyboard and 005 accesses a CRT in a System 2200 configuration.

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



For input as follows:

1) System commands.
2) Immediate Mode statements.
3) Program text entry.

For input as follows:

1) Data for INPUT statements.
2) Data for KEYIN statements.
3) Data for MAT INPUT statements.

For I/O operations:

1) BACKSPACE*
2) DATALOAD
3) DATALOAD BT
4) DATARESAVE
5) DATASAVE
6) DATASAVE BT
7) LOAD
8) REWIND*
9) SAVE
10) SKIP*
11) $GIO
12) $IF ON

For disk or diskette operations:

1) COPY
2) DATALOAD BA
3) DATALOAD DA
4) DATALOAD DC
5) DATALOAD DC OPEN
6) DATASAVE BA
7) DATASAVE DA
8) DATASAVE DC
9) DATASAVE DC CLOSE
10) DATASAVE DC OPEN
11) DBACKSPACE
12) DSKIP
13) LIMITS
14) LOAD DA
15) LOAD DC
16) MOVE
17) MOVE END
18) SAVE DA
19) SAVE DC
20) SCRATCH
21) SCRATCH DISK
22) VERIFY

For output as follows:

1) Data from Immediate Mode PRINT or HEXPRINT stmts.
2) Literal string messages from INPUT statements.
3) Question marks when the system is awaiting INPUT-class data.
4) Echo of data received for INPUT or MAT INPUT stmts.
5) Colons when the system is ready for CI-class input.
6) Error message codes.
7) STEP mode printouts.
8) TRACE mode printouts.
9) Other system messages.

For output as follows:

1) Data from Program Mode PRINT or HEXPRINT stmts.
2) Data from PRINTUSING and associated Image stmts.
3) Data from MAT PRINT stmts.

For output as follows:

1) Program text from LIST commands.
2) Disk data from LIST DC statements.

For output as follows:

1) Graphs and labels from PLOT statements.

*Not for Model 2207A applications.

APPENDIX C
BASIC LANGUAGE SYNTAX AND TERMS

The following syntax is used to denote the components in a general form of a statement:

1. Uppercase letters (A through Z) or words must be written in an actual statement exactly as shown in a general form.

2. Lowercase letters or words represent items for which specific information is to be substituted in an actual statement. Sometimes hyphens join lowercase words (or words and numbers) to signify single items.

3. Vertically stacked items represent alternatives, only one of which is to be selected.

4. When stacked items are enclosed in braces, { }, one item must be specified. The braces are not included in an actual statement.

5. When single or stacked items are enclosed in brackets, [ ], the items are optional and may be omitted. The brackets are not included in an actual statement.

6. The following characters must be written as shown in a general form, unless otherwise indicated by a note:

comma	,
equal sign	=
parentheses	()
pound-sign	#
slash	/
double quotation marks	"

7. When an ellipsis, ..., follows an item, the item may be repeated many times successively in an actual statement.

8. Blanks, inserted for readability in a general form, are not required. Wang systems ignore blanks in an actual statement unless the blanks are embedded in quotation marks.

9. The sequential order of the components in a general form must be preserved when writing an actual statement.

The following terms occur in one or more of the general forms in this manual:

alpha array designator	-	Any alphanumeric array name followed by closed parentheses, e.g., A$(), M6$().
alpha variable	-	Any alphanumeric scalar variable, e.g., B$, C2$; any element of a one- or two-dimensional alphanumeric array, e.g., D$(4), X$(15,7); or any string-function, e.g., STR(Y$, 5, 3).

array designator	- Any alphanumeric or numeric array name followed by closed parentheses, e.g., Y$(), X().
expression	- Any combination of numeric variables, digits, arithmetic operations, and built-in mathematical functions, e.g., 2 * (X↑ 3 + SIN(Y)).
numeric array designator	- Any numeric array name followed by closed parentheses, e.g., X(), Z5().
numeric variable	- Any numeric scalar variable, e.g., Y, Z2; or any element of a one- or two-dimensional numeric array, e.g., C(7), P3(1), D(4,9).

APPENDIX D
FUNDAMENTAL I/O STATEMENTS

INPUT
KEYIN
MAT INPUT
PRINT
PRINTUSING
MAT PRINT

NOTE:

The operations INPUT, KEYIN, MAT INPUT, PRINT, PRINTUSING, and MAT PRINT have built-in features which may be suitable for program control of non-Wang devices interfaced to a Wang CPU. The six operations are described in detail in this appendix to assist readers in determining their appropriateness for particular applications. See Appendices E and F for descriptions of other I/O operations.

General Form:

```
    INPUT ["literal string",] variable [, variable]...
```

Purpose

The INPUT statement controls data input from one device and sends system-generated information to another device. The statement has many special features, some designed to prompt an operator when input is required via a keyboard during program operation; however, the statement can be used to request data (without operator intervention) from Wang peripherals other than a keyboard, or from some non-Wang devices interfaced to a 2200 Series CPU.

The primary purpose of an INPUT statement is to receive data from the device whose address is currently selected for operations in the I/O-class INPUT. Prior to data reception, however, the literal string message (if any) and a question mark are sent to the device whose address is currently selected for the operations in the I/O-class CO (console output); then, the system awaits data input. As each character is received, an echo is sent to the CO device.

General Features

The argument list of an INPUT statement can contain several variables representing memory locations into which data is to be stored. Numeric variables, alphanumeric variables, string (STR) functions, and specific array elements can be used as arguments, but not array designators; therefore, the maximum character length per argument is 64. To ensure proper storage of input data without character loss, a long message or data stream received in response to an INPUT statement must contain data-separator commas at intervals not exceeding 65 characters and carriage-return-characters at intervals not exceeding 191 characters.

Data is stored temporarily in the CPU buffer and not processed for transfer to variables in the argument list until a carriage-return-character is received.

When processing begins, several features are implemented, e.g., a data validity check is made before data is transferred to a numeric variable.

If the data delimited by a carriage-return-character does not satisfy all the variables in the argument list, the input procedure resumes automatically and does not terminate until all variables are satisfied or a carriage-return-character is received without data.

Special Features

Execution of an INPUT statement is equivalent to the following steps:

1. If the statement contains a literal string message, the message is sent to the address currently selected for CO-class operations (or the primary address 005 if no selection has been made).

2. A question mark character, followed by a space character is sent to the CO address (usually to prompt an operator that the system is awaiting data).

3. The CPU enables the device controller whose address is currently selected for INPUT-class operations (or the primary address 001 if no selection has been made) and awaits an incoming character.

4. As soon as a character is received from the input device (via its controller board in the CPU chassis), the character is transferred to the CPU buffer for temporary storage; then an echo is sent to the CO address. The echo-procedure disables the INPUT address, enables the CO address, outputs the echo character, and usually re-enables the INPUT address to await the next character.

NOTE:

The following character codes produce special action:

a) Codes $(08)_{16}$ and $(5F)_{16}$ are interpreted as backspace instructions and are not stored. Each code effectively removes the previously buffered character.

b) Code $(5C)_{16}$ is interpreted as a line erase instruction and is not stored. The code effectively removes all the currently-buffered characters.

c) Code $(0D)_{16}$ is interpreted as a signal to interrupt data reception and process data currently in the CPU buffer. See Step 7.

5. A count is maintained of the number of characters currently stored in the CPU buffer, and Step 4 is repeated until a carriage-return-character $(0D)_{16}$ is stored in the buffer or until the buffer contains 191 characters (whichever occurs first).

6. If the buffer count reaches 191 without an $(0D)_{16}$ code, the CPU disables the INPUT address, enables the CO address, and outputs an error message (ERR 45). The buffered data is ignored (not stored in memory). The count is reset to zero and execution returns to Step 3. If an $(0D)_{16}$code is received, execution goes to Step 7.

7. After an $(0D)_{16}$ code reaches the CPU buffer, the CPU disables the INPUT address and begins to process the buffered data.

8. The system checks the position of the argument list "pointer" to determine whether the next receiving variable is numeric or alphanumeric. Also, the system looks ahead in the buffered data until a data separator is found.

NOTE:

The following rules apply to data separators:

a) A carriage-return code $(0D)_{16}$ can serve as a data separator in addition to serving as a signal to process data currently in the CPU buffer.

b) A comma $(2C)_{16}$ serves as a data separator if not embedded in a pair of quotation marks which qualify as data delimiters.

c) Quotation marks, i.e., $(22)_{16}$ the code for double quotation marks and $(27)_{16}$ the code for single quotation marks qualify as data delimiters if they occur in matched pairs as follows:

- the first quotation mark is the first nonblank character in the buffer or the first nonblank character following the last recognized data separator comma, and
- the second quotation mark is followed by a comma or a carriage-return code.

If the second quotation mark is missing, error message (ERR 39) is sent to the CO address, the buffered data is ignored, the argument list pointer does not move, and execution returns to Step 2. If the second quotation mark is present but the separator is missing, error message (ERR 29) is sent, the buffered data is ignored, and execution returns to Step 2.

d) Commas which qualify as data separators and quotation marks which qualify as delimiters are not stored in memory. Other commas and quotation marks are treated as data to be stored.

9. After a data separator is located, the data preceding the separator is processed for transfer to the receiving variable:

a) If the receiving variable is numeric, the system tests the data with respect to legal BASIC language numeric formats. If the data contains illegal characters, error message (ERR 29) is sent to the CO address, the buffered data is ignored, the argument list pointer does not move, and execution returns to Step 2. On the other hand, legal data is converted to Wang's internal numeric format for storage in the receiving variable.

b) If the receiving variable is alphanumeric, no data validity check is made; the characters are transferred to memory sequentially until the location is filled or the data separator is reached (whichever occurs first). If the location in memory is too small, any excess input characters are ignored. If the location is larger than the data, trailing space characters $(20)_{16}$ are transferred. In particular, if the receiving variable is a string (STR) function, only the specified byte-positions receive data from the buffer. If the input data is enclosed in single quotation marks, uppercase codes are converted to lowercase codes before the data is stored.

NOTE:

The codes $(00)_{16}$ and $(7F)_{16}$ are ignored. Any code above $(7F)_{16}$ may give unpredictable results.

If no data precedes a data separator carriage-return, execution is terminated.

10. After data is transferred to a receiving variable, the argument list pointer moves to the next variable, if any. If there is another variable and the last data separator was a comma, execution returns to Step 8; if the last separator was a carriage-return code, execution returns to Step 2. On the other hand, if the list contains no other receiving variables, execution of the INPUT statement is terminated.

Examples

To illustrate how commas and double-quotation-marks are interpreted by an INPUT statement, five brief examples are included here. Identical input is used for each example.

No DIM statement or SELECT statement is included in any of the two-line programs in Examples D1 through D5. If the programs are entered and run after the system has been Master Initialized, the keyboard is selected automatically as the input device, and the dimension for each alphanumeric variable is 16 characters (the default value).

The PRINT statement is used in each example to find out what characters are stored in memory for each variable named in the INPUT statement. However, the PRINT statement presents the data in a zoned or a packed format depending upon whether a comma or a semicolon follows each variable. A HEXPRINT statement could be used to determine what characters are stored in memory.

Examples D1 through D5 are presented here as hardcopy of the CRT display produced when the short programs are entered, run, and given input especially prepared to demonstrate the System 2200 INPUT-statement-processing-procedure. To duplicate any example, first enter the program text (i.e., lines 10 and 20). Then enter the command RUN by touching the RUN key followed by the EXECUTE key. After the question mark appears to indicate the system is awaiting data, enter the input shown in the fourth line. The output shown in the fifth line appears automatically.

Example D1

```
:10 INPUT A$, B$
:20 PRINT A$, B$
:RUN
? "AB,CD",M"PQ,RS"
AB,CD           M"PQ
```

The first comma in the input data stream is considered by the system to be part of the literal string intended for storage in the variable A$ memory location. The second comma is recognized as a separator and is not stored. The first double-quotation-mark of the next pair of double-quote characters is not the first nonblank character after the data-separator-comma; therefore, the expression "PQ,RS" is not recognized as a literal string. Instead, M"PQ is assigned to B$ because the comma after the letter Q is recognized as another data separator. Since the INPUT statement requires data for only two variables, the remaining data RS" is ignored. Output appears in zoned format because a comma is used after A$ in the PRINT statement.

Example D2

```
:10 INPUT A$,B$,C$
:20 PRINT A$,B$,C$
:RUN
? "AB,CD",M"PQ,RS"
AB,CD           M"PQ            RS"
```

By adding a third variable to the INPUT (and PRINT) statements in Example D2, the input data previously ignored in Example D1 is stored in C$.

Example D3

```
:10 INPUT A$,B$
:20 PRINT A$;B$
:RUN
? "AB,CD",M"PQ,RS"
AB,CDM"PQ
```

By using a semicolon in Line 20, instead of the comma used in Example D1, the output appears in packed rather than zoned format.

Example D4

```
:10 INPUT A$,B$,C$
:20 PRINT A$;B$;C$
:RUN
? "AB,CD",M"PQ,RS"
AB,CDM"PQRS"
```

By using semicolons in Line 20, instead of the commas used in Example D2, the output appears in packed format.

Example D5

```
:10 INPUT A$,B$,C
:20 PRINT HEX(22);A$;HEX(22);HEX(2C);B$;HEX(2C);C
:RUN
? "AB,CD",XYZ,125.3
"AB,CD",XYZ, 125.3
```

By using the code HEX(22) for each double-quotation-mark and HEX(2C) for each data-separator-comma, the PRINT statement in Line 20 structures a data stream which duplicates the input stream (except for the leading space allotted for a sign when numeric input is assigned to the numeric variable C). A trailing space is allotted also when the value is printed.

```
General Form:

     KEYIN alpha-variable, line-number-1, line-number-2,

where:

alpha-variable =      a specified alphanumeric varible, a
                      string function, or a particular
                      element of an alphanumeric array.

line-number-1  =      a specified line for a conditional
                      branch.

line-number-2  =      a second (or the same) specified line
                      for a conditional branch.
```

Purpose

The KEYIN statement (which can be used only in the Program Mode) checks the character-ready condition of the device controller whose address is currently selected for the I/O-class parameter INPUT. Then, one of the following actions occurs, depending upon the character-ready condition:

. Not ready - program execution proceeds to the next statement.

. Ready with a character, other than a special function code - the character is stored in the first byte-position of the specified variable, and program execution branches to line-number-1.

. Ready with a special function code, i.e., one of the 32 codes ranging from $(00)_{16}$ through $(1F)_{16}$ - the code is stored in the first byte of the specified variable, and program execution branches to line-number-2.

Unlike the INPUT statement, the KEYIN statement provides no inherent display capability, i.e., no character echo is sent to the currently selected CO address.

Test Program

The following program can be used to demonstrate the conditional branches of the KEYIN statement. In the program, Line 20 is executed repeatedly by an endless loop. Each time no character is found in the buffer on the keyboard controller, the Line 30 message is displayed. However, touching alpha or numeric keys on the keyboard produces the Line 50 message, while touching Special Function Keys produces the Line 70 message.

Statements	Remarks
10 DIM A$1	Dimension A$ for one character.
20 KEYIN A$, 50, 70	Scan the keyboard for input.
30 PRINT "******"	
40 GOTO 20	
50 PRINT "---to---"	
60 GOTO 20	
70 PRINT "S.F. KEY"	
80 GOTO 20	

By including the following line:

```
15 SELECT INPUT xyy
```

where xyy is replaced by the "input" address of an interface controller, the program can be used to demonstrate KEYIN operations for a device plugged into the interface controller.

NOTE:

For applications not involving single character input, the $IF ON statement offers advantages over a KEYIN statement. A $IF ON statement can check the device-ready condition of an input or an output address, and branch to a specified line if a ready condition is sensed.

General Form:

$$\text{MAT INPUT} \left\{ \begin{array}{l} \text{numeric-array-name [(}d_1\text{ [,}d_2\text{])]} \\ \text{alpha-array-name [(}d_1\text{[, }d_2\text{])[length]]} \end{array} \right\} \text{ [,...]}$$

where:

d = expression specifying a new dimension, $1 \le d_1, d_2 \le 255$ and $d_1 * d_2 \le 4096$ (default $d_1 = d_2 = 10$ if no COM or DIM statement defines the dimensions).

length = expression specifying the dimensioned length of each alphanumeric array element, $1 \le \text{length} \le 64$ (default length = 16 bytes).

Purpose

The MAT INPUT statement controls data input from one device and sends system-generated information to another device using a procedure very similar to the INPUT statement. However, unlike the INPUT statement, the MAT INPUT statement receives values for storage in numeric and/or alphanumeric arrays. The syntax requires array names only, not array designators; closed parentheses are implied and must be omitted when specifying the argument list. Any arrays previously dimensioned in a COM or DIM statement can be redimensioned in a MAT INPUT statement.

Arrays are filled element-by-element, row-by-row sequentially. Input values must be separated by commas or carriage-returns at intervals not exceeding the element size to prevent data loss. If leading space characters or commas are to be treated as part of an alphanumeric value, the data should be enclosed in double quotation marks and followed by a data separator. Data is temporarily buffered in the CPU until a carriage-return character is received. For further details, read the discussion of the INPUT statement presented in this appendix.

```
General Form:

    PRINT print-element [t print-element]...[t]

where:
                  ⎧literal string        ⎫
                  ⎪alphanumeric variable ⎪
print-element  =  ⎨HEX function          ⎬ = output data source
                  ⎪expression            ⎪
                  ⎪TAB parameter         ⎪
                  ⎩null                  ⎭

            t =  ⎧comma    ⎫ = element separator and format indicator
                 ⎩semicolon⎭
```

Purpose

The PRINT statement outputs data in zoned, packed, or mixed format depending upon whether commas, semicolons, or both are used as print-element separators. For a packed format (indicated by semicolons), data from a particular print-element is followed immediately by data from the next print-element. For a zoned format (indicated by commas), data from a particular print-element is followed by enough space characters to complete a zone before data from the next print-element begins. (Output zones represent as many groups of 16 characters each as possible for the currently selected line length, e.g., there are five zones in an 80 character line length; the default line length is 64 characters; the maximum length is 255.)

Features

A PRINT statement can be executed in the Immediate Mode or the Program Mode; however, different I/O-class parameters are associated with each operation as follows:

- In the Program Mode, a PRINT statement outputs data to the address currently selected for the I/O-class PRINT. If no address has been selected for PRINT operations, the system uses the primary address 005.

- In the Immediate Mode, a PRINT statement outputs data to the address currently selected for the I/O-class CO (console output). If no address has been selected for CO operations, the system uses the primary address 005.

Data output is affected by the following general controls:

- The data format for each type of print-element is fixed by the system.

- Extra space characters (blanks) are output automatically, when necessary, to satisfy a zoned format.

- A character count is maintained, and a carriage-return character, followed by either a line-feed or a null character (depending upon the device type code in the address), is output automatically whenever one of the following conditions arises:

  - the current character count equals the line length (unless device type = 4),

  - the system looks ahead and finds that the sum of the current character count and the characters to be output for the next print element exceeds the line length, or

  - when no punctuation follows the last print-element.

  (Note: The character count is reset to zero when a carriage return character is output.)

Data Formats

Data formats for each type of print-element are determined by the following rules:

1. Literal Strings -

   Data is output character-by-character, including any blanks, exactly as shown within the double quotation marks delimiting a string, or, if single quotation marks delimit a string, uppercase characters are replaced by lowercase characters in the output.

2. Alphanumeric Variables -

   The characters stored in the specified variable are output, except for trailing space characters which are ignored. If a variable contains all space characters, one space character is output. Note: To output every character in an alphanumeric variable, a string function should be used as a print-element, e.g., STR(A$,1) outputs every character in A$, beginning with the first.

3. HEX Functions -

   A HEX function, e.g., HEX(41582C35), defines each character by a pair of hexdigits. Such functions can be used to output any 8-bit codes. Data is output byte-by-byte sequentially until every specified character is sent. Note: the look-ahead feature is not implemented before the system outputs HEX function data; therefore, a system-generated carriage return and line feed (or null) character sequence is inserted automatically when the character count equals the line length - then the count is reset to zero and data output continues.

4. Null Print-elements -

   If a print-element is omitted (i.e., a second element separator immediately follows a previous one), the system outputs one null character, $(00)_{16}$.

5. TAB Parameters

First, the expression in a TAB parameter is evaluated and truncated to an integer (which must be < 256). Then, one of two possible actions occurs:

- If the current character-count is less than the TAB value, space characters are output until the count equals the TAB value.

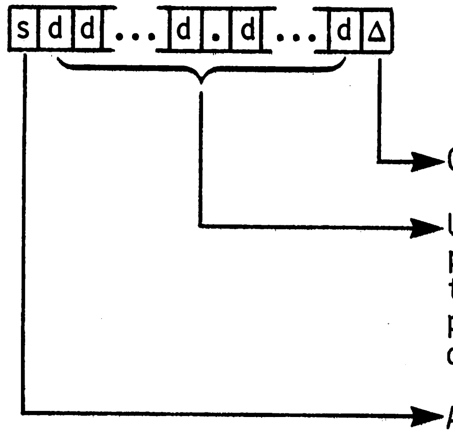
- If the current character-count is greater than or equal to the TAB value, the TAB parameter is ignored.

Note: Codes whose first hexdigit is zero, do not increment the character count, e.g., a null character $(00)_{16}$, a carriage-return character $(0D)_{16}$, a line-feed character $(0A)_{16}$, etc.

6. Expressions -

First, the expression is evaluated; then, the value determines whether a fixed point or a floating point format is used for data output. Numeric formats for the value of an expression (or a single numeric variable) are defined as follows, where $|Q|$ represents the absolute value of the print-element:

<u>Fixed Point Format</u> if $.1 \leq |Q| < 10^{+13}$



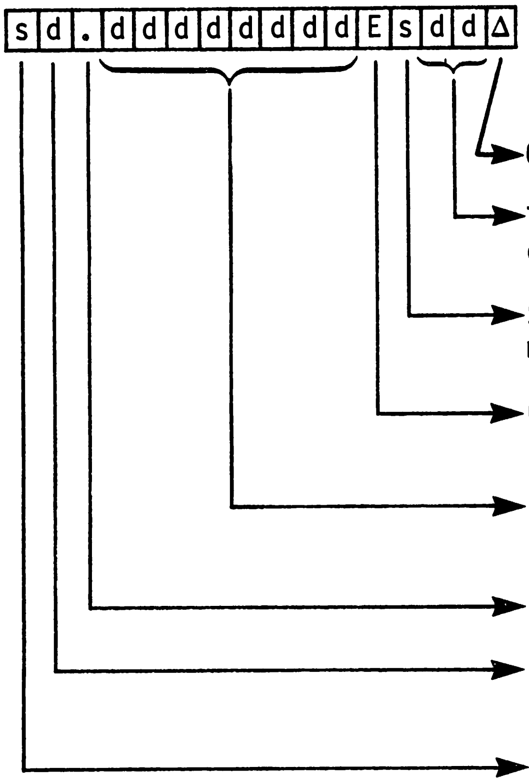
One trailing blank = $(20)_{16}$.

Up to 13 digits and an optional decimal point in ASCII code. Leading and trailing zeros are omitted. The decimal point = $(2E)_{16}$ is in the proper position or omitted if Q is an integer.

A blank = $(20)_{16}$ if $Q \geq 0$, or a minus sign = $(2D)_{16}$ if $Q < 0$.

(Note: The number of characters in a fixed point format varies from a minimum of three to a maximum of 16, including the trailing blank.)

<u>Floating Point Format</u> if $10^{-99} < Q < .1$, or $10^{+13} \leq Q < 10^{+100}$



One trailing blank = $(20)_{16}$.

Two exponential digits in ASCII code.

Sign of exponent: plus = $(2B)_{16}$, minus = $(2D)_{16}$.

Character denoting exponential format, E = $(45)_{16}$.

Eight digits including trailing zeros in ASCII code.

Decimal point = $(2E)_{16}$.

Non-zero leading digit before decimal point (scientific notation).

Sign of value: blank = $(20)_{16}$ if $Q \geq 0$, minus = $(2D)_{16}$ if $Q < 0$.

(Note: The number of characters in a floating point format is always 16, including the trailing blank.)

General Procedure

Upon execution, a PRINT statement processes the specified print-elements sequentially.

For each print-element, data is formatted according to the rules applicable to the element type. After data output is completed for the print-element currently being processed, one of the following actions occurs:

- If a semicolon follows the current print-element, the next element is processed immediately.
- If a comma follows the current print-element, the system determines how many space characters are needed to complete the current zone (i.e., to move a printing device to the proper position for subsequent output). The necessary number of space characters are output before the next element is processed.
- If no punctuation follows the current print-element, the system outputs a carriage-return character automatically (followed by a line-feed or null character).

Effect of Device Type Codes on PRINT Statement Output

Printing devices do not have identical characteristics. For example, some devices automatically supply a line-feed character each time a carriage-return character is received. Some devices automatically supply a carriage-return and a line-feed when received data exceeds their physical carriage width.

However, if an appropriate device type code is used when an address is selected for the I/O-class parameter PRINT, the System 2200 adjusts its PRINT statement processing procedure as shown in the following table:

Effect of Device Type Code on PRINT Statement Output

Code	System Action
0	Supplies a line-feed character automatically after each system-generated carriage-return (CR) character. Therefore, device type "0" is appropriate when outputting data to a device which does not supply its own line-feed (LF) character after receiving a CR. (See Code "4" also.)
2	Supplies a null character automatically after each system-generated CR. Therefore, device type "2" is appropriate when outputting data to a device which supplies its own LF after receiving a CR.
4	Suppresses the character count and thereby suppresses any CR normally generated when the character count equals the currently selected line length. Supplies a LF after any other system-generated CR. Therefore, device type "4" is appropriate when outputting data to a device which supplies its own CR/LF when the physical carriage width is exceeded.

(Note: The address 005 with device type "0" is standard for the CRT. The address 215 with device type "2" is standard for Wang printers. With the address 215, the printer supplies its own LF after receiving a CR/NULL character sequence; however, if the address 015 is used, the printer supplies its own LF after receiving a CR/LF character sequence -- thereby causing double spaced hardcopy.)

General Form:

```
PRINTUSING line-number [,print-element] [t print-element]...[;]
```

where:

line-number = the number of the image (%) statement defining the output format.

print-element = $\left\{\begin{array}{l}\text{literal string}\\ \text{alphanumeric variable}\\ \text{expression}\end{array}\right\}$ = output data source.

t = $\left\{\begin{array}{l}\text{comma}\\ \text{semicolon}\end{array}\right\}$ = element separator.

General Form:

% $\left\{\begin{array}{c}l\\ f\end{array}\right\}$ $\left\{\begin{array}{c}l\\ f\end{array}\right\}$ ...

where:

l = spaces or a literal string (not enclosed in quotation marks), with no pound-sign symbols (#) or colons (:).

f = a data format specification of the following form:

$\left\{\begin{array}{c}+\\ -\\ \$\end{array}\right\}$ # [,] [#...] [.] [#...] [↑↑↑↑]

Purpose

A PRINTUSING statement, in conjunction with a referenced Image (%) statement, can be used to output print-element data interspersed with alphanumeric text, according to the data format specifications and text in the image.

Since PRINTUSING and % statements are executable only in the Program Mode, data is output to the address last selected for the I/O-class parameter PRINT. If no address has been selected for PRINT operations, the primary address 005 is used.

Data Formats

Data format specifications for alphanumeric output should be composed only of # symbols (as many symbols as the number of characters to be output). Data format specifications for numeric output should be composed of # symbols (one for each digit) and optional symbols such as +, -, $, ., and ↑↑↑↑. (See the output rules for expressions.)

Output Rules

The following rules apply to print-element output for PRINTUSING and Image statements:

1. Alphanumeric Variables

   Data stored in the variable is output one character per # symbol beginning with the leftmost symbol in the format specification. If the variable is shorter than the format, space characters are added to satisfy the number of # symbols. If the variable is longer than the format, the excess characters are ignored.

2. Literal Strings as Print Elements

   Data enclosed in quotation marks is output one character for each # symbol in the format, beginning with the leftmost symbol, until each symbol is satisfied. If the literal string is shorter than the format, space characters are added until the format is satisfied. If the string is longer than the format, the string is truncated (i.e., the excess characters are ignored).

3. Expressions

   Numeric data is output with respect to the format specification as follows:

   a) If the format begins with a plus (+) sign, the actual sign of the value (+ or -) is output immediately preceding the first significant digit.

   b) If the format begins with a minus (-) sign, a space character (a blank) is output for a positive value or a minus sign (-) is output for a negative value immediately preceding the first significant digit.

   c) If the format begins with a dollar ($) sign, a $ character is output immediately preceding the first significant digit.

   d) If no plus, minus, or dollar sign is shown at the beginning of the format, a minus sign (-) is output for a negative value and the length of the format is increased by one character automatically.

   e) Any commas in the integer portion of a numeric format specification are output in the order shown in the format if a significant digit has been output immediately before their occurrence; otherwise, a space character is output instead.

General Procedure

Print-elements are processed sequentially. Data corresponding to the first print-element specified in the PRINTUSING statement is edited to conform to the first format specification in the Image statement. Image statement text, if any, is output before or after the edited print-element data as required by the image.

The process of pairing print-elements and format specifications continues until all elements are processed. If the number of print-elements in the PRINTUSING statement exceeds the number of format specifications in the referenced Image statement, the Image statement is reused from the beginning for the remaining print-elements. Each time an image statement is reused, one of the following actions is implemented:

- If a semicolon precedes the particular print-element which exceeds the format specification and thereby initiates reuse of the image, one space character is output prior to data output.

- If a comma precedes the print-element which initiates reuse of the image, a carriage-return character followed by a line feed or a null character (depending on the device type code) is output prior to data output.

MAT PRINT

General Form:

```
    MAT PRINT array-name [t array-name]...[t]
```

where:

$$t = \left\{\begin{matrix}\text{comma} \\ \text{semicolon}\end{matrix}\right\} = \text{argument-list separator and format indicator}$$

Purpose

The MAT PRINT statement outputs data in zoned, packed, or mixed format depending upon whether commas, semicolons, or both are used as separators in the argument list.

Data from each two-dimensional array is output row-by-row. The first element of a row always starts at the beginning of a new output line and successive elements are output in zoned format unless the array name is followed by a semicolon, in which case, the elements are output in packed format. For alphanumeric arrays, the zone length is set equal to the dimensioned length of the array elements (rather than 16). As many output lines as required for the elements of one row are formatted before data corresponding to the next row begins. Space characters and carriage-return characters are inserted in the output data automatically, as required, to achieve the required format.

Data from each one-dimensional array is output in a column vector format, i.e., one element per line.

APPENDIX E
SPECIAL PURPOSE I/O STATEMENTS

DATALOAD
DATALOAD BT
DATASAVE
DATASAVE BT
LOAD
SAVE

NOTE:

The statements in this appendix can be used to read/punch information on some paper tape devices interfaced to a System 2200 via a Model 2207A controller. It is possible that the statements can be used to transfer data to or from other devices also.

DATALOAD

General Form:

$$\text{DATALOAD} \quad \begin{bmatrix} \#f, \\ /xyy, \end{bmatrix} \text{ argument-list}$$

where:

f = An indirect address -- a file number (1,2,3,4,5 or 6) to which the appropriate three-hexdigit address has been assigned by a SELECT statement.

xyy = An absolute address -- a three-hexdigit address with x representing the appropriate device type code and yy representing the appropriate controller address.

argument-list = $\begin{Bmatrix} \text{variable} \\ \text{array-designator} \end{Bmatrix}$,. . .

Note: If neither a file number nor an absolute address is specified, the default address is the one last selected for the I/O class TAPE.

Purpose

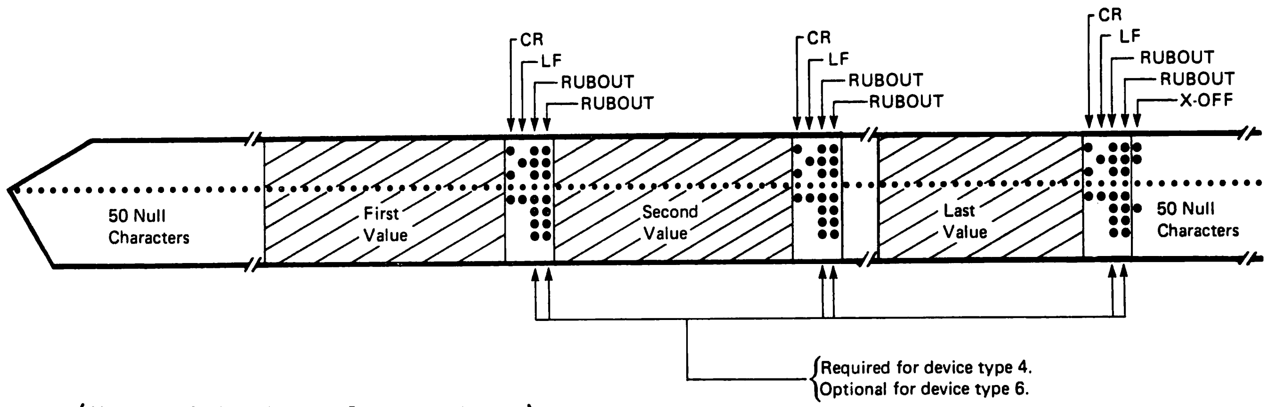
With the device type code equal to 4 or 6, the DATALOAD statement can read values from a punched tape and assign the values sequentially to the variables in the argument list. Arrays are filled element-by-element, row-by-row. Only the first seven channels of a punched tape are read; the eighth channel is interpreted as a "0" bit whether punched or not.

If device type 4 is used in the address, an X-ON character $(11)_{16}$ is transmitted automatically to start the tape reader unit; an X-OFF character $(13)_{16}$ is transmitted to stop the unit when reading is completed. These automatic codes are needed for control of Teletype or Teletype-like read/punch devices.

If device type 6 is used, X-ON and X-OFF characters are not transmitted to start and stop a tape reader unit. Such characters are not required to control a unit like Wang's Model 2203 Punched Tape Reader.

Punched Tape Format

To be read by a DATALOAD statement, a punched tape must conform to the following format:



(Note: 8th channel not shown)

Values should be punched in ASCII code since the eighth channel is interpreted as a "0" bit, whether punched or not. Values should be separated by the four-character sequence consisting of a CR (carriage-return) LF (line-feed) RUBOUT RUBOUT. The two rubout characters are required if device type 4 is used; they are optional if device type 6 is used. All other rubout characters and nonpunched frames on a tape are ignored when the tape is read, whether the device type is 4 or 6.

Tapes punched using DATASAVE statements automatically conform to the format required by DATALOAD statements. However, to ensure satisfactory retrieval of data, the DATALOAD argument list used to read a tape should be identical to the DATASAVE argument list used to punch a tape. (Two argument lists are identical if the number, type, and sequential order of the variables match -- names may differ.)

Tapes not conforming to the required format for DATALOAD statements should be read using DATALOAD BT statements.

Special Features

Values read from a punched tape are assigned sequentially to variables in the argument list under the following conditions:

- arrays are filled element-by-element, row-by-row,
- both numeric and alphanumeric values can be assigned to alphanumeric variables, and
- only valid System 2200 fixed and floating point numbers can be assigned to numeric variables since an automatic check is made before data is stored in a numeric variable.

Values are read from a tape until all variables in the argument list are satisfied or until an end-of-file (an X-OFF character) is read. If an X-OFF character is read before the argument list is satisfied, the remaining variables retain their current values and an end-of-file condition is set.

By programming an IF END THEN statement after a DATALOAD statement (and specifying the initial line number of a subroutine), special messages can be printed or special action taken in response to an end-of-file condition. During execution of an IF END THEN statement, a branch is made to the specified line number only if an end-of-file condition currently exists; also, the condition is reset (removed). In programs without an IF END THEN statement, the end-of-file condition (if set) is reset when the system encounters another DATALOAD statement or initiates program execution.

Examples of Valid Syntax

```
DATALOAD /41D, A1$(), X, Y
DATALOAD #3, N(), A$, STR(B$, I, J)
DATALOAD X,Y,A$,B$
```

NOTES:

1. When using a DATALOAD statement to control input from a device interfaced via a dual-address controller, the yy-digits in the device address for the operation should be replaced by the "output address".

2. A DATALOAD statement with device type 6 may prove useful for control of interfaced devices other than tape reader units if the incoming data values are separated by carriage-return characters.

General Form:

$$\text{DATALOAD BT [R][parameter-set]} \begin{bmatrix}\text{\#f,}\\ \text{/xyy,}\end{bmatrix} \begin{Bmatrix}\text{alpha-variable}\\ \text{alpha-array-designator}\end{Bmatrix}$$

where:

R = Read in reverse direction. (Can be used with device type 6 only.)

$$\text{parameter-set} = \left(\big[\text{N = expression}\big], \left[\text{L} = \begin{Bmatrix}\text{hh}\\ \text{alpha-variable}\end{Bmatrix}\right], \left[\text{S} = \begin{Bmatrix}\text{hh}\\ \text{alpha-variable}\end{Bmatrix}\right] \right)^{*}$$

N = Number of characters to be read (each character = 1 byte = 8 bits). The expression is evaluated and truncated to an integer which must be ≥ 1.

L = The leader code, i.e., a code to be ignored by the system until a different code is read. The code can be specified by a pair of hexdigits (represented by hh above) or by an alphanumeric variable whose first character defines the code.

S = The stop code, i.e., a code instructing the system to stop reading data. The code can be specified by a pair of hexdigits (represented by hh above) or by an alphanumeric variable whose first character defines the code.

f = An indirect address -- a file number (1,2,3,4,5 or 6) to which the appropriate three-hexdigit address has been assigned by a SELECT statement.

xyy = An absolute address -- a three-hexdigit address with x representing the appropriate device type code and yy representing the appropriate controller address.

Note: If neither a file number nor an absolute address is specified, the default address is the one last selected for the I/O-class TAPE.

* When more than one parameter is specified in the closed parentheses, commas must separate the parameters, e.g., (N = 100, L = 00).

Purpose

With the device type code equal to 4 or 6, the DATALOAD BT statement can read program text or data from a punched tape and store the information in the specified alphanumeric variable or array. All eight channels of the tape are read.

If device type 4 is used in the address, an X-ON character $(11)_{16}$ is transmitted automatically to start the tape reader unit; an X-OFF character $(13)_{16}$ is transmitted to stop the unit when reading is completed. These automatic codes are needed for control of Teletype or Teletype-like read/punch devices.

If device type 6 is used in the address, an initialization character of the following form is transmitted automatically:

$(xxxx0000)_2$	if the parameter R is omitted, or
$(xxxx0001)_2$	if the parameter R is included in the statement.

In either case, the four high-order bits are indeterminate. Such codes are needed to activate a device like Wang's Model 2203 Punched Tape Reader. In particular, the code with low-order bit "1" activates "forward reading," and the code with low-order bit "0" activates "reverse reading".

If the parameter L is used in a DATALOAD BT statement to define a leader code, the code is ignored when the tape is read, i.e., all characters equal to the specified code are ignored until the first character not equal to the code is recognized.

The following considerations should influence the values used in an actual statement <u>only if</u> device type 4 is used:

1. For termination by count (the N parameter), the system normally sends out the X-OFF character after N-2 characters have been read; therefore, N should be ≥ 3. If N=1 (or 2), the next 2 (or 1) characters may be lost.

2. If reading is terminated by filling the variable or array, the number of characters in the variable or array should be >3.

3. If a stop character is encountered, the stop character and the next two characters are read; then the tape stops.

<u>No Special Tape Format Required</u>

The DATALOAD BT statement permits punched tapes in any format to be read by the system. If the data is not in a form directly usable by the system, the data can be retrieved from the alphanumeric variable or array and converted using other BASIC language statements which perform bit/byte manipulation or conversion operations.

<u>Examples of Valid Syntax</u>

```
DATALOAD BT /41D, A$
DATALOAD BT (L = FF, S = 0D) #2, A$()
DATALOAD BT (N = 100) X$()
DATALOAD BT (N = 20 , L = 00, S = A$) A1$()
```

NOTES:

1. When using a DATALOAD BT statement to control input from a device interfaced via a dual-address controller, the yy-digits in the device address for the operation should be replaced by the "output address".

2. A DATALOAD BT statement with device type 6 may prove useful for control of interfaced devices other than tape reader units.

DATASAVE

General Form:

```
DATASAVE   [#f,  ] (OPEN "name"   )
           [/4yy,] <END            >
                   (argument-list )
```

where:

f = An indirect address -- a file number (1,2,3,4,5 or 6) to which the appropriate three-hexdigit address has been assigned by a SELECT statement.

4yy = An absolute address -- a three-hexdigit address with device type code equal to 4 and yy representing the appropriate controller address.

OPEN = A parameter indicating that leader code is to be punched (50 null characters).

name = Any 1 to 8 character name (required by the syntax but not used).

END = A parameter indicating that an X-OFF character and trailer code are to be punched.

```
argument-
    list = (literal string   )
           (alpha variable   ),...
           (expression       )
           (array designator )
```

Note: If neither a file number nor an absolute address is specified, the default address is the address last selected for the I/O-class TAPE.

Purpose

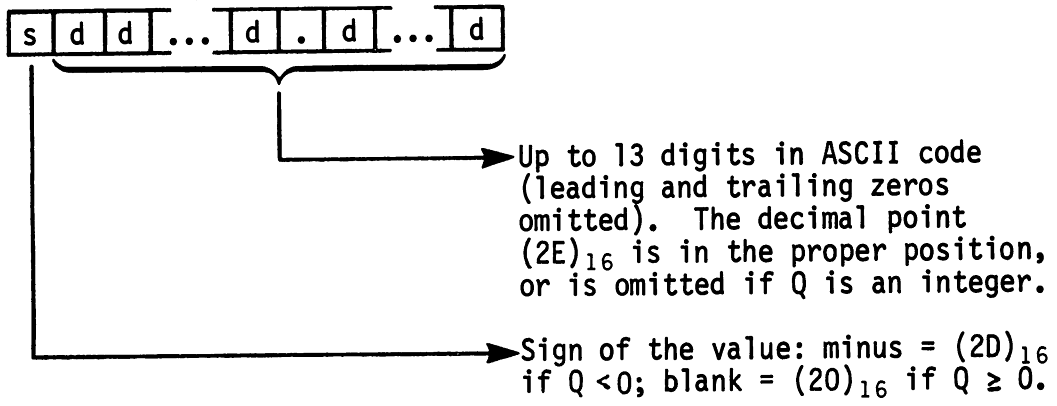
With the device type code equal to 4, the DATASAVE statement can be used to punch data files on paper tape. Three versions of the statement perform distinct phases of a data file punching operation. A statement with the parameter OPEN (followed by a dummy name in quotation marks) punches leader code, i.e., 50 null characters. A statement with an argument list sequentially punches data values corresponding to the argument list. A statement with the parameter END punches a file termination code, i.e., an X-OFF character, followed by trailer code consisting of 50 null characters.

Upon execution, a DATASAVE statement with an argument list automatically inserts the four-character-sequence CR (carriage-return) LF (line feed) RUBOUT RUBOUT wherever needed to separate values. Array values are punched element-by-element, row-by-row sequentially with the four-character-sequence separating the elements.

When a value corresponding to an alphanumeric variable is punched, any trailing space characters stored in memory are omitted.

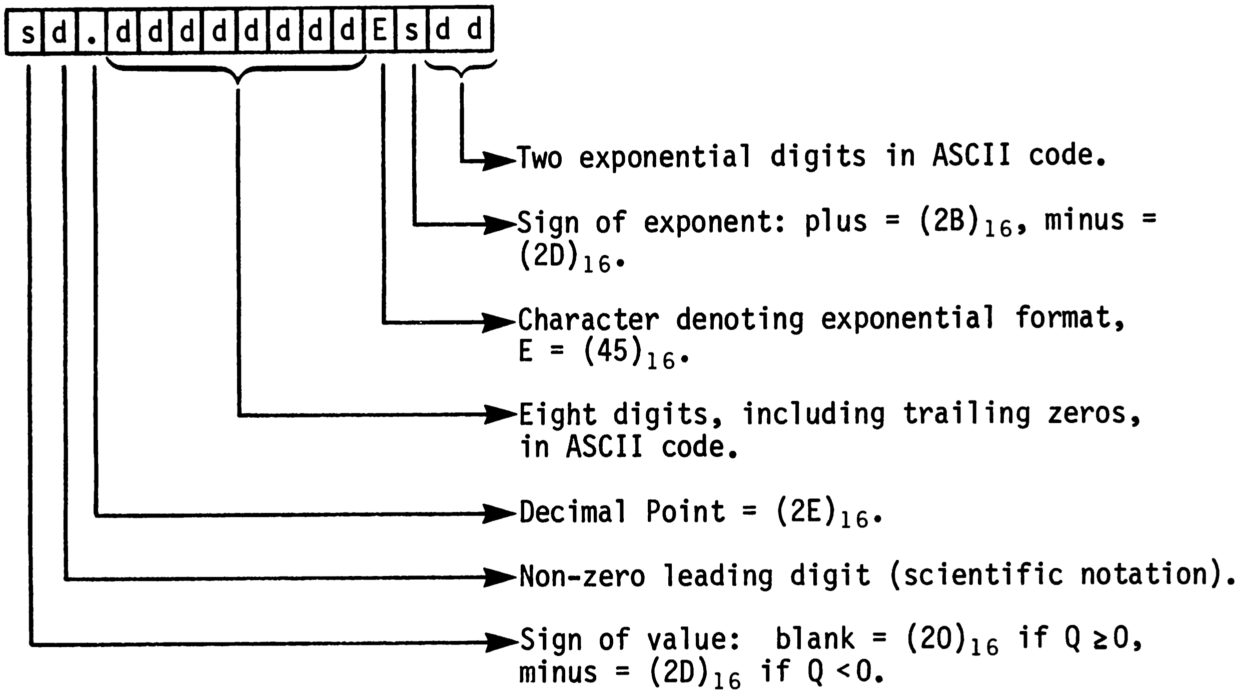
When a value corresponding to a numeric variable is punched, the value is automatically converted from Wang's internal numeric format to either a fixed or a floating point format, depending upon the magnitude of the value (denoted by $|Q|$ in the following diagrams).

Fixed Point Format if $.1 \leq |Q| < 10^{+13}$


s d d ... d . d ... d
Up to 13 digits in ASCII code (leading and trailing zeros omitted). The decimal point $(2E)_{16}$ is in the proper position, or is omitted if Q is an integer.
Sign of the value: minus = $(2D)_{16}$ if $Q < 0$; blank = $(20)_{16}$ if $Q \geq 0$.

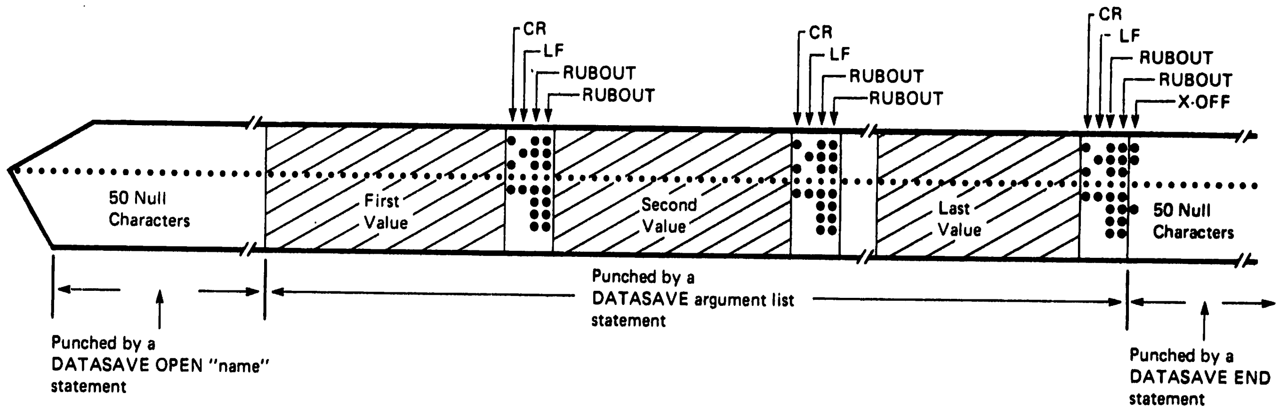
(Note: A fixed point value may have a minimum of two characters or up to 15 characters.)

Floating Point Format if $10^{-99} \leq |Q| < .1$, or if $10^{+13} \leq |Q| < 10^{+100}$


s d . d d d d d d d d E s d d
Two exponential digits in ASCII code.
Sign of exponent: plus = $(2B)_{16}$, minus = $(2D)_{16}$.
Character denoting exponential format, E = $(45)_{16}$.
Eight digits, including trailing zeros, in ASCII code.
Decimal Point = $(2E)_{16}$.
Non-zero leading digit (scientific notation).
Sign of value: blank = $(20)_{16}$ if $Q \geq 0$, minus = $(2D)_{16}$ if $Q < 0$.

(Note: A floating point value always has 15 characters.)

Punched Tape Format:


CR
LF
RUBOUT
RUBOUT
CR
LF
RUBOUT
RUBOUT
CR
LF
RUBOUT
RUBOUT
X-OFF
50 Null Characters
First Value
Second Value
Last Value
50 Null Characters
Punched by a DATASAVE OPEN "name" statement
Punched by a DATASAVE argument list statement
Punched by a DATASAVE END statement

(Note: 8th channel on punched tape not shown)

<u>Examples of Valid Syntax</u>

```
DATASAVE OPEN "TTY"
DATASAVE N(), A$, X,Y,Z
DATASAVE END
DATASAVE /41D, STR(A$,I,J), HEX(FAFB)
DATASAVE #5, A$()
DATASAVE X, Y, A$
```

NOTE:

When using a DATASAVE statement to control output to a device interfaced via a dual-address controller, the yy-digits in the device address for the operation should be replaced by the "output address".

General Form:

```
    DATASAVE BT [#f,  ] {alpha-variable         }
                [/4yy,] {alpha-array-designator }
```

where:

f = An indirect address -- a file number (1,2,3,4,5, or 6) to which the appropriate three-hexdigit address has been assigned by a SELECT statement.

4yy = An absolute address -- a three-hexdigit address with device type code equal to 4 and yy representing the appropriate controller address.

Note: If neither a file number nor an absolute address is specified, the default address is the address last selected for the I/O-class TAPE.

Purpose

With the device type code equal to 4, the DATASAVE BT statement can be used to punch on tape the value currently stored in a specified alphanumeric variable or the values currently stored in an alphanumeric array. No control information is inserted between values. Data in any 8-bit code format can be punched.

If the interfaced tape punch equipment is capable of turning its punch unit on upon receipt of an X-ON code and turning the unit off upon receipt of an X-OFF code, the hexadecimal equivalent of these codes can be transmitted under program control prior to and following execution of a DATASAVE BT statement by using PRINT statements. For example, PRINT HEX(11) transmits the X-ON code, and PRINT HEX(13) transmits the X-OFF code.

Examples of Valid Syntax

```
DATASAVE BT /41D, B1$
DATASAVE BT #2, A$()
DATASAVE BT Q$()
```

NOTE:

When using a DATASAVE BT statement to control output to a device interfaced via a dual-address controller, the yy-digits in the device address for the operation should be replaced by the "output address".

LOAD

General Forms for the LOAD Command and LOAD statement:

```
Command:                   LOAD   [#f  ]
                                  [/4yy]

Statement (Program Mode Only):   LOAD  [#f  ][,line-1[,line-2]]
                                       [/4yy]
```

where:

f = An indirect address -- a file number (1,2,3,4,5 or 6) to which the appropriate three hexdigit address has been assigned by a SELECT statement.

4yy = An absolute address -- a three-hexdigit address with device type code equal to 4 and yy representing the appropriate controller address.

line-1 = A number denoting the starting line of text to be deleted before reading new text into memory. If not specified, the first line of the text is implied. (After new text is loaded, execution automatically continues at "line-1"; an error occurs if the new text does not contain such a line.)

line-2 = A number denoting the final line of text to be deleted before loading the next text. If not specified, the last line of the text is implied.

1. The LOAD command differs in procedure from the LOAD statement. (LOAD is the only System 2200 verb which appears in a command and a statement. By definition a command is nonprogrammable.) See the functions of the LOAD command and LOAD statement as described below.
2. If neither a file number nor an absolute address is specified, the default address is the address last selected for the I/O-class TAPE.

Purpose (LOAD Command)

With the device type code equal to 4 or 6, the LOAD command reads program text from a punched tape and appends the text to the program currently in memory. Thus the command permits additions to a current program or, if executed after a CLEAR command, permits entry of a new program. The new program is not executed until a separate RUN command is entered.

Purpose (LOAD Statement)

The LOAD statement produces an automatic combination of events equivalent to the following sequence of commands:

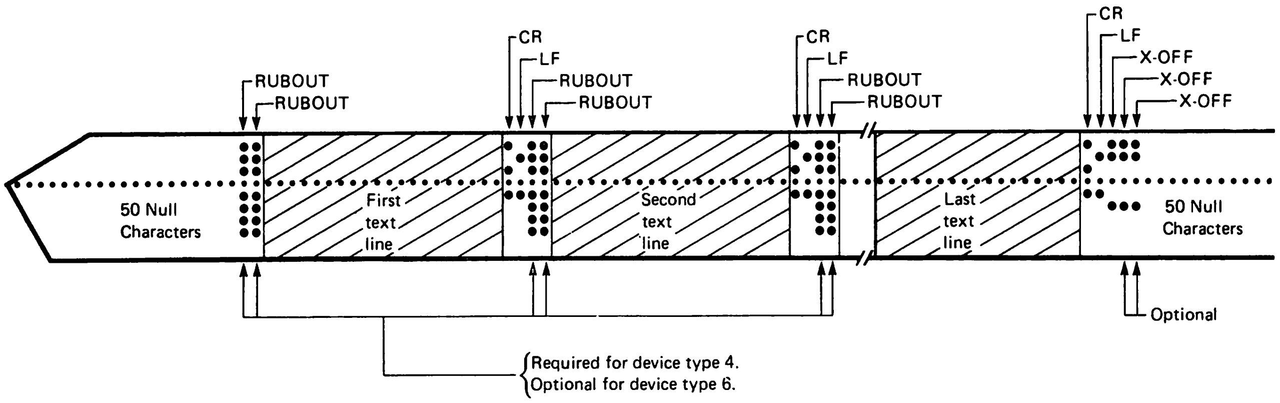
STOP -	Stop execution of current program.
CLEAR P -	Remove program text defined by [,line-1[,line-2]]
CLEAR N -	Remove noncommon variables only.
LOAD -	Load new program text.
RUN -	Run the new program beginning with [line-1].

The line numbers are optional. In particular, if only line-1 is specified, the current program is deleted starting with the specified line and continuing through the remaining text. If no line numbers are specified, the entire program is deleted, and execution of the newly loaded program begins with its lowest line number.

The LOAD statement, if included in a multistatement line, must be the last statement. A LOAD statement must not appear in a subroutine or within a FOR/NEXT loop; if so, an error occurs when the RETURN statement or the NEXT statement is encountered.

With the device type code equal to 4 or 6, the LOAD statement (executable in Program Mode only) permits segmented, chained, or overlayed programs to be run automatically, without operator intervention if the segments are properly formatted on a punched tape.

Required Tape Format for LOAD Commands and LOAD Statements



(Note: 8th channel not shown)

Text lines must be punched in ASCII character code and separated by the four-character-sequence CR (carriage-return) LF (line-feed) RUBOUT RUBOUT if device type code 4 is used. The two RUBOUT characters are optional if device type code 6 is used. The program must be terminated by at least one X-OFF character.

A LOAD command or statement reads only the first seven channels of a paper tape; the eighth channel is always interpreted as a "0" bit. Nonpunched frames and RUBOUT characters are ignored.

Examples of Valid Syntax

```
   Commands:    LOAD
                LOAD #1
                LOAD /41D

 Statements:    100 LOAD
                250 LOAD #2
                400 LOAD /41D, 100
```

NOTE:

When using a LOAD command or statement to control input via a dual-address controller, the yy-digits in the device address for the operation should be replaced by the "output address".

General Form:

```
SAVE  [#f  ]  [,line-1 [,line-2]]
      [/4yy]
```

where:

f = An indirect address -- a file number (1,2,3,4,5 or 6) to which the appropriate three hexdigit address has been assigned by a SELECT statement.

4yy = An absolute address -- a three hexdigit address with device type code equal to 4 and yy representing the appropriate controller address.

line-1 = A number denoting the starting line of program text to be saved. If not specified, the first line of text is implied.

line-2 = A number denoting the final line of text to be saved. If not specified, the last line of text is implied.

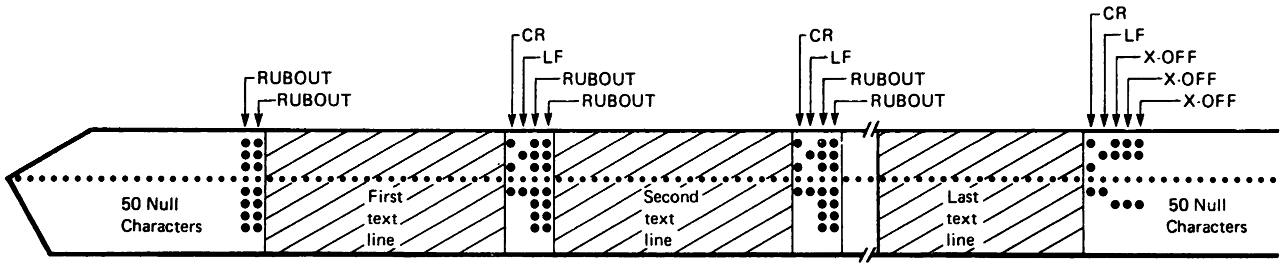
Note: If neither a file number nor an absolute address is specified, the default address is the address last selected for the I/O-class TAPE.

Purpose

With the device type code equal to 4, the SAVE command punches program text on a paper tape. Line numbers for a particular command are optional. If only line-1 is specified, a portion of the current program is saved (starting with the specified line and continuing through the remainder of the text). If no line numbers are specified, the entire text is saved.

Tape Format

Upon execution, a SAVE command punches a tape in the following format:



(Note: 8th channel on punched tape not shown)

Beginning with two RUBOUT characters, the tape contains each text line punched in ASCII character codes and separated from the next line by the four-character-sequence CR (carriage-return) LF (line-feed) RUBOUT RUBOUT. The last text line is followed by three X-OFF characters.

Examples of Valid Syntax

```
SAVE
SAVE #3
SAVE/41D
SAVE/41D, 100, 200
SAVE #5, 400
```

NOTE:

When using a SAVE command to control output to a device interfaced via a dual-address controller, the yy-digits in the device address for the operation should be replaced by the "output address".

APPENDIX F
CUSTOMIZED I/O STATEMENTS

$GIO

NOTE:

Customizing a $GIO operation to suit a particular device and application is similar to machine language programming. The $GIO statement has a "general input/output" format executable within the framework of the high-level BASIC language; however, a microcommand sequence must be directly or indirectly specified in a statement to define the particular operation to be performed. Seventeen categories of microcommands are available for use with the $GIO statement.

General Form:

```
    $GIO  [comment] [#f  ]  (arg-1,arg-2) [arg-3]
                    [/xyy]
```

where:

comment = A character string, similar to a remark, identifying the operation, e.g., WRITE, READ, CHECK READY; the comment is ignored by the system.

f = An indirect address -- a file number (1,2,3,4,5 or 6) to which the appropriate three-hexdigit address has been assigned by a SELECT statement.

xyy = An absolute address -- a three-hexdigit address with "0" as the recommended value for x, and with yy representing the appropriate controller address.

arg-1 = A customized microcommand sequence defining the particular operation; the sequence can be specified directly by a set of microcommands (four-hexdigit-codes), or indirectly by a variable representing a stored sequence.

arg-2 = An alphanumeric variable representing the error/status/general-purpose registers, i.e., a multi-purpose memory area where special characters and error/status information are stored in particular byte-positions called registers.

arg-3 = An alphanumeric variable, array designator, or modified array designator representing the data buffer (required only for a microcommand sequence which includes a multicharacter input or output microcommand).

Note: If neither a file number nor an absolute address is specified, the default address is the address last selected for the I/O-class TAPE.

Purpose

The $GIO statement executes a custom-tailored signal sequence defined by a specified microcommand sequence. The statement is ideally suited for support of non-Wang devices and instruments interfaced to a System 2200 via one of Wang's interface controllers.

Features

$GIO microcommands are similar to machine language codes. Each microcommand is a four-hexdigit-code representing a fundamental operation (usually multi-step) which can be used as a "building block" when constructing a $GIO operation. Seventeen categories of microcommands are available with many choices in each category. The capability to customize an operation is almost limitless since individual microcommands can implement one or more diverse functions such as the following:

- setting a delay condition applicable to subsequent output of each character
- setting a timeout condition applicable to subsequent sensing of device-ready signals and data input
- disabling a previously enabled delay or timeout condition
- storing special characters in particular registers
- comparing registers and setting error flags or terminating the operation
- outputting immediate or indirect (stored) characters with or without awaiting an acknowledge or echo character
- outputting a device address to deselect a currently selected device and select a different device
- inputting single or multicharacter data with or without echoing or verifying the received data
- sending strobes to request each character during a multicharacter input operation
- setting CPU ready signals
- awaiting device ready signals
- terminating an operation by count or special-character-comparison
- calculating, sending and/or saving the longitudinal redundancy check (LRC) character for a multicharacter output or input operation.

Several microcommands arranged in a particular sequence in the arg-1 component of a $GIO statement can produce a complex operation or a relatively simple one. Sometimes only one microcommand may be needed for a particular operation.

The registers (byte-positions) in the variable specified as the arg-2 component of a $GIO statement are reserved for storage of the following types of information:

- an indirect character for a single-character output operation or for comparison during a single-character input with verify operation
- a two-byte value defining a delay or a timeout interval
- a special character defining a termination condition for a multicharacter input operation
- an acknowledge or echo character received after a single character output operation

- an LRC character calculated during a multicharacter input or output operation
- a two-byte binary count of the total number of transferred characters (whether stored or not) when a buffer overflow occurs during multicharacter input
- error/status flags (stored on a bit-by-bit basis in Register 8) indicating such conditions as buffer overflow, LRC error, echo/verify error, compare error, timeout exceeded, termination (by count, special character, or ENDI-level).

Valid Microcommands

The General I/O Instruction Set Reference Manual, furnished with each central processing unit which includes the $GIO statement, contains tables of all the valid microcommands with a description of the signal sequence associated with each microcommand. The tables are too lengthy to reproduce in this appendix.

Example

The following example illustrates some capabilities of the $GIO statement.

```
$GIO/01D (0202 0300 4011 1221 7105 4000 711D 1200 A000 4013, R$) B$()
```

Microcommand	Function
0202	Store the character $(02)_{16}$ in Register 2 (the second byte of R$).
0300	Store the character $(00)_{16}$ in Register 3.
4011	Send the character $(11)_{16}$ to the currently selected address, i.e., 1D.
1221	Set a delay condition equal to 50 microseconds multiplied by the two-byte binary value stored in Registers 2 and 3.
7105	Deselect the current address and select the address 05 (the CRT).
4000	Send the null character $(00)_{16}$ to the current address (now the CRT).
711D	Deselect the current address and select the address 1D.
1200	Disable the delay specified by the microcommand 1221.
A000	Output each character stored in the buffer B$(), using the following sequence: WR = wait for a ready signal from the enabled device. DATAOUT/OBS = send the next character with an OBS strobe. LEND = the LRC End sequence specified by hexdigit h_4. Since $h_4 = 0$, the sequence is "None".
4013	Send the character $(13)_{16}$ to the currently selected address, i.e., 1D.

The microcommand sequence above includes some operations which might be useful for output of data to a punch tape unit. The microcommand 4011 sends an X-ON character to the unit. The microcommands 1221, 7105 and 4000 send a null character to the CRT (after a delay of 25.6 seconds) to allow the punch motor time to reach a specified condition. The delay is disabled before the multicharacter output operation (represented by A000) begins. Finally, the microcommand 4013 sends an X-OFF character to the unit.

NOTE:

Before an appropriate microcommand sequence can be chosen, the application and hardware requirements must be defined and related to the inherent features of the available microcommands. Therefore, the example given in this appendix should not be used to control a particular punch tape unit until the microcommands and the hardware requirements are understood.

APPENDIX G - 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.

THREE BITS (HIGH ORDER) $b_7\ b_6\ b_5$	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1	7-BIT BINARY NOTATION
FIRST HEX DIGIT (HIGH ORDER) HEX a_1	0	1	2	3	4	5	6	7	$b_4\ b_3\ b_2\ b_1$
0	NUL	DLE	SP	0	@	P	\`	p	0 0 0 0
1	SOH	DC1	!	1	A	Q	a	q	0 0 0 1
2	STX	DC2	"	2	B	R	b	r	0 0 1 0
3	ETX	DC3	#	3	C	S	c	s	0 0 1 1
4	EOT	DC4	$	4	D	T	d	t	0 1 0 0
5	ENQ	NAK	%	5	E	U	e	u	0 1 0 1
6	ACK	SYN	¢	6	F	V	f	v	0 1 1 0
7	BEL	ETB	' (apos)	7	G	W	g	w	0 1 1 1
8	BS	CAN	(	8	H	X	h	x	1 0 0 0
9	HT	EM	)	9	I	Y	i	y	1 0 0 1
A	LF	SUB	*	:	J	Z	j	z	1 0 1 0
B	VT	ESC	+	;	K	[	k	{	1 0 1 1
C	FF	FS	, (comma)	<	L	\	l	\	1 1 0 0
D	CR	GS	– (dash)	=	M	]	m	}	1 1 0 1
E	SO	RS	. (period)	>	N	↑	n	~	1 1 1 0
F	SI	US	/	?	O	(underline)	o	DEL	1 1 1 1

CHARACTERS — SECOND HEX DIGIT a_2 (LOW ORDER) — FOUR BITS (LOW ORDER)

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
		SUB	Substitute
LF	Line Feed	ESC	Escape
VT	Vertical Tabulation	FS	File Separator
FF	Form Feed	GS	Group Separator
CR	Carriage Return	RS	Record Separator
SO	Shift Out	US	Unit Separator
SI	Shift In	DEL	Delete

APPENDIX H
SETTING ADDRESS SWITCHES ON THE MODEL 2207A CONTROLLER

Two 8-pole address switches, RCV and XMT, are located on the chip side of the controller board. In each switch, 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 H-1 with a grid added for reference purposes only.

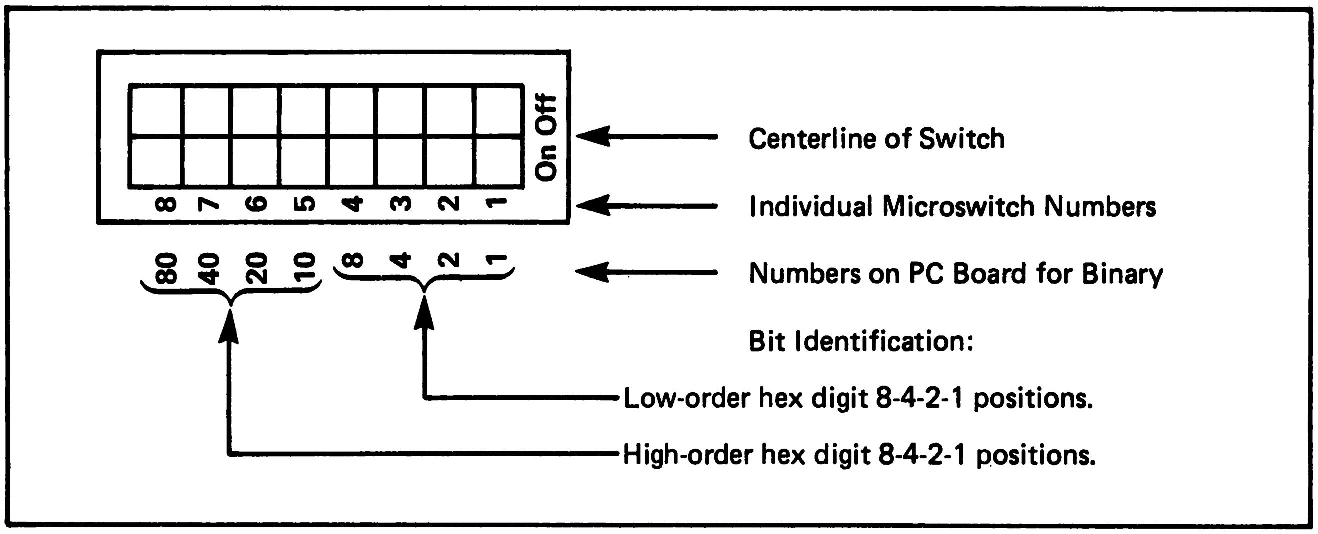

On Off
8 7 6 5 4 3 2 1
80 40 20 10 8 4 2 1
Centerline of Switch
Individual Microswitch Numbers
Numbers on PC Board for Binary
Bit Identification:
Low-order hex digit 8-4-2-1 positions.
High-order hex digit 8-4-2-1 positions.

Figure H-1. Frame for an Address Switch

Each of the eight microswitches (not shown in Figure H-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 is 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 2207A unit, the standard address codes are x19 for the RCV switch and x1D for the XMT switch. The standard addresses for a second unit in a configuration are X1A and X1E. 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 H-1.

The last two hexdigits of a 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 should be turned OFF. When a bit is one, the corresponding microswitch should be turned ON (see Table H-1).

Table H-1. Standard Address Switch Settings for Model 2207A Controllers

Unit*	I/O Channel	Hexadecimal	Binary
1	Input (RCV) Output (XMT)	19 1D	00011001 00011101
2	Input (RCV) Output (XMT)	1A 1E	00011010 00011110
3	Input (RCV) Output (XMT)	1B 1F	00011011 00011111

*If a unit is installed for hookup of a Teletype or Teletype-like device to serve as a console input/output unit, the address switches should be set as follows:

Input (RCV) = $(01)_{16}$ = $(00000001)_2$
Output(XMT) = $(05)_{16}$ = $(00000101)_2$

APPENDIX I
MODEL 2207A SPECIFICATIONS

Size of Controller Board

Length.	14 in. (35.6 cm)
Depth	6 in. (15.2 cm)
Width	1 in. (2.5 cm)

Weight

3 lb (1.4 kg)

Power Requirements

Supplied by the CPU.

Connector

Receives a 25-pin RS-232-C compatible male plug.

Switches

Internal:	Two device address switches on printed circuit board.
External:	Five transmission rate switches: 110, 150, 300, 600 and 1200 baud.

Asynchronous Transmission Format

ASCII mode:	1 start bit, 7 data bits, even parity bit, 2 stop bits.
Binary mode:	1 start bit, 8 data bits, 2 stop bits.

Transmission Rate

Switch selectable: 110, 150, 300, 600 or 1200 baud.

Special Features

In ASCII mode, automatically decodes a Teletype BREAK signal into a HALT/STEP signal and decodes an ESC (Escape) signal into a RESET signal.

Standard Warranty Applies

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.

WANG The completed order form should be mailed to:
WANG LABORATORIES, INC.
Supplies Division
51 Middlesex St.
No. Chelmsford MA 01863

To Order by Phone, Call:
(800)225-0234
From Mass., Hawaii, and Alaska
(617)256-1400
TELEX 951-743

Order Form for Wang Manuals and Documentation

① Customer Number (If Known)

② Bill To: _______ Ship To: _______

③ Customer Contact: ( ) ( ) Phone _______ Name _______ ④ Date _______ Purchase Order Number _______

⑤ Taxable Yes ☐ No ☐ ⑥ Tax Exempt Number ⑦ Credit This Order to A Wang Salesperson Please Complete _______ Salesperson's Name _______ Employee No. _______ RDB No.

⑧ Document Number	Description	Quantity	⑨ Unit Price	Total Price

⑩ _______ Authorized Signature _______ Date

☐ Check this box if you would like a free copy of the **Corporate Publications Literature Catalog** (700-5294)

Sub Total	
Less Any Applicable Discount	
Sub Total	
Local State Tax	
Total Amount	

Ordering Instructions

1. If you have purchased supplies from Wang before, and know your Customer Number, please write it here.
2. Provide appropriate Billing Address and Shipping Address.
3. Please provide a phone number and name, should it be necessary for WANG to contact you about your order.
4. Your purchase order number and date.
5. Show whether order is taxable or not.
6. If tax exempt, please provide your exemption number.
7. If you wish credit for this order to be given to a WANG salesperson, please complete.
8. Show part numbers, description and quantity for each product ordered.
9. *Pricing extensions and totaling can be completed at your option; Wang will refigure these prices and add freight on your invoice.*
10. Signature of authorized buyer and date.

Wang Supplies Division Terms and Conditions

1. **TAXES** — Prices are exclusive of all sales, use, and like taxes.
2. **DELIVERY** — Delivery will be F.O.B. Wang's plant. Customer will be billed for freight charges; and unless customer specifies otherwise, all shipments will go best way surface as determined by Wang. Wang shall not assume any liability in connection with the shipment nor shall the carrier be construed to be an agent of Wang. If the customer requests that Wang arrange for insurance the customer will be billed for the insurance charges.
3. **PAYMENT** — Terms are net 30 days from date of invoice. Unless otherwise stated by customer, partial shipments will generate partial invoices.
4. **PRICES** — The prices shown are subject to change without notice. Individual document prices may be found in the Corporate Publications Literature Catalog (700-5294)
5. **LIMITATION OF LIABILITY** — In no event shall Wang be liable for loss of data or for special, incidental or consequential damages in connection with or arising out of the use of or information contained in any manuals or documentation furnished hereunder.

WANG

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 16 NO. CHELMSFORD, MA.

POSTAGE WILL BE PAID BY ADDRESSEE

WANG LABORATORIES, INC.
Supplies Division
c/o Order Entry Dept.
M/S 5511
51 Middlesex St.
No. Chelmsford, MA 01863

Fold

WANG

Customer Comment Form

Title 2207A I/O INTERFACE CONTROLLER USER MANUAL

Publications Number 700-3364

Help Us Help You . . .

We've worked hard to make this document useful, readable, and technically accurate. Did we succeed? Only you can tell us! Your comments and suggestions will help us improve our technical communications. Please take a few minutes to let us know how you feel.

How did you receive this publication?

- ☐ Support or Sales Rep
- ☐ Wang Supplies Division
- ☐ From another user
- ☐ Enclosed with equipment
- ☐ Don't know
- ☐ Other ____________

How did you use this Publication?

- ☐ Introduction to the subject
- ☐ Classroom text (student)
- ☐ Classroom text (teacher)
- ☐ Self-study text
- ☐ Aid to advanced knowledge
- ☐ Guide to operating instructions
- ☐ As a reference manual
- ☐ Other ____________

Please rate the quality of this publication in each of the following areas.

	EXCELLENT	GOOD	FAIR	POOR	VERY POOR
Technical Accuracy — Does the system work the way the manual says it does?	☐	☐	☐	☐	☐
Readability — Is the manual easy to read and understand?	☐	☐	☐	☐	☐
Clarity — Are the instructions easy to follow?	☐	☐	☐	☐	☐
Examples — Were they helpful, realistic? Were there enough of them?	☐	☐	☐	☐	☐
Organization — Was it logical? Was it easy to find what you needed to know?	☐	☐	☐	☐	☐
Illustrations — Were they clear and useful?	☐	☐	☐	☐	☐
Physical Attractiveness — What did you think of the printing, binding, etc?	☐	☐	☐	☐	☐

Were there any terms or concepts that were not defined properly? ☐ Y ☐ N If so, what were they? ____________

After reading this document do you feel that you will be able to operate the equipment/software? ☐ Yes ☐ No ☐ Yes, with practice

What errors or faults did you find in the manual? (Please include page numbers) ____________

Do you have any other comments or suggestions? ____________

Name ____________ **Street** ____________

Title ____________ **City** ____________

Dept/Mail Stop ____________ **State/Country** ____________

Company ____________ **Zip Code** ________ **Telephone** ____________

Thank you for your help.

All comments and suggestions become the property of Wang Laboratories, Inc.

WANG

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 16 LOWELL, MA

POSTAGE WILL BE PAID BY ADDRESSEE

WANG LABORATORIES, INC.
CHARLES T. PEERS, JR., MAIL STOP 1369
ONE INDUSTRIAL AVENUE
LOWELL, MASSACHUSETTS 01851

Fold

Cut along dotted line.

WANG

ONE INDUSTRIAL AVENUE
LOWELL, MASSACHUSETTS 01851
TEL. (617) 459-5000
TWX 710-343-6769, TELEX 94-7421

Printed in U.S.A.
700-3364
10-82-4C