WANG

Synchronous/Asynchronous Communications Controller User Manual

(Model 2228C, Model 2228B, or Option 62B)

T. Olson - 1213

Synchronous/ Asynchronous Communications Controller User Manual

(Model 2228C, Model 2228B, or Option 62B)

© Wang Laboratories, Inc., 1978

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

Disclaimer of Warranties and Limitation of Liabilities

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

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

PREFACE

Chapter 1 discusses the installation and dual capabilities of Wang's Synchronous/Asynchronous Communications Controller. The chapter is written primarily for persons having overall responsibility for Wang systems and their use.

Chapters 2 and 3 independently discuss the two types of communications supported by the controller. These chapters, written primarily for programmers, may be omitted by anyone planning to use Wang-developed software to communicate with another system. Operating instructions are provided with the software acquired for use with the controller.

CONTENTS

TABLES

FIGURES

CHAPTER 1

GENERAL INFORMATION

1.1 THE CONTROLLER

Wang's Synchronous/Asynchronous Communications Controller is available in two physically different but operationally equivalent versions. The Model 2228B version is a double-card controller attached to a mounting bar for plug-in compatibility with the I/O slots in some 2200 Series Central Processing Units (e.g., the 2200T, 2200VP, and 2200MVP). The Option 62B version has no mounting bar and is configured for Wang systems with a console-housed central processor (e.g., the PCS-II). There is also a Model 2228C version which can support Wang's 3275 BSC Emulator in addition to all the software supported by the Model 2228B controller. A copy of this manual is provided with each Synchronous/Asynchronous Communications Controller acquired for a Wang system.

The controller has its own microprocessor, read only memory, and random access memory, including multicharacter input and output buffers. Thus, separate tasks related to data transmission and reception can be performed by the controller and the central processor concurrently--if appropriate software and a suitable modem are used in conjunction with the controller.

Microcode defining synchronous protocol tasks to be performed by the controller is produced only by Wang Laboratories and made available in terminal emulation software packages which also contain a set of program modules defining tasks to be performed by the central processor. When the Wang-developed synchronous communications software is used, all necessary microcode is automatically loaded into the controller.

Built-in microcode supports asynchronous communications tasks; therefore, asynchronous communications software may be user-developed or acquired from Wang Laboratories. Information necessary for development of asynchronous application programs is presented in Chapter 3.

Wang's Synchronous/Asynchronous Communications Controller is a two-in-one controller whose dual capabilities are suited to a wide variety of communications applications. Hence, to avoid confusion, readers of this manual should selectively concentrate on the information related to either synchronous or asynchronous communications, not both types of information at once. Furthermore, once a Wang system is equipped with the communications controller, appropriate software, and a compatible modem, anyone planning to transmit or receive data may wish to go immediately to the manual provided with the software.

1.2 INSTALLATION

Installation of a communications controller is the responsibility of a Wang Customer Engineer. If a controller arrives as an addition to existing equipment, call the Wang Customer Engineer to schedule the installation. Do not attempt to install the controller--any unauthorized installation voids the warranty.

A 25-pin EIA (Electronic Industries Association) RS-232-C, CCITT V.24 compatible connector is located on the Model 2228B mounting bar or on the back panel of a system having a console-housed central processor. A 12-foot (3.6m) cable is supplied with the controller. The connector and the cable facilitate hookup of a modem.

After the controller is inspected, diagnostically checked, and installed, one end of the supplied cable is plugged into the connector attached to the controller. Then, if a synchronous or asynchronous type modem has already been installed near the Wang system, the other end of the cable is plugged into the modem. However, installation of a modem is not the responsibility of a Wang Customer Engineer, except in those cases where a "null modem" is to be used. (See Section 1.3.)

1.3 MODEM CONSIDERATIONS

A modem (i.e., a modulator/demodulator) is needed for communications with remotely located computers or digital equipment since data signals from a computer must be converted (modulated) into a range of frequencies suitable for transmission over telephone lines. Similarly, data signals received via telephone lines must be demodulated before transfer to a computer. (See Figure 1-1.)

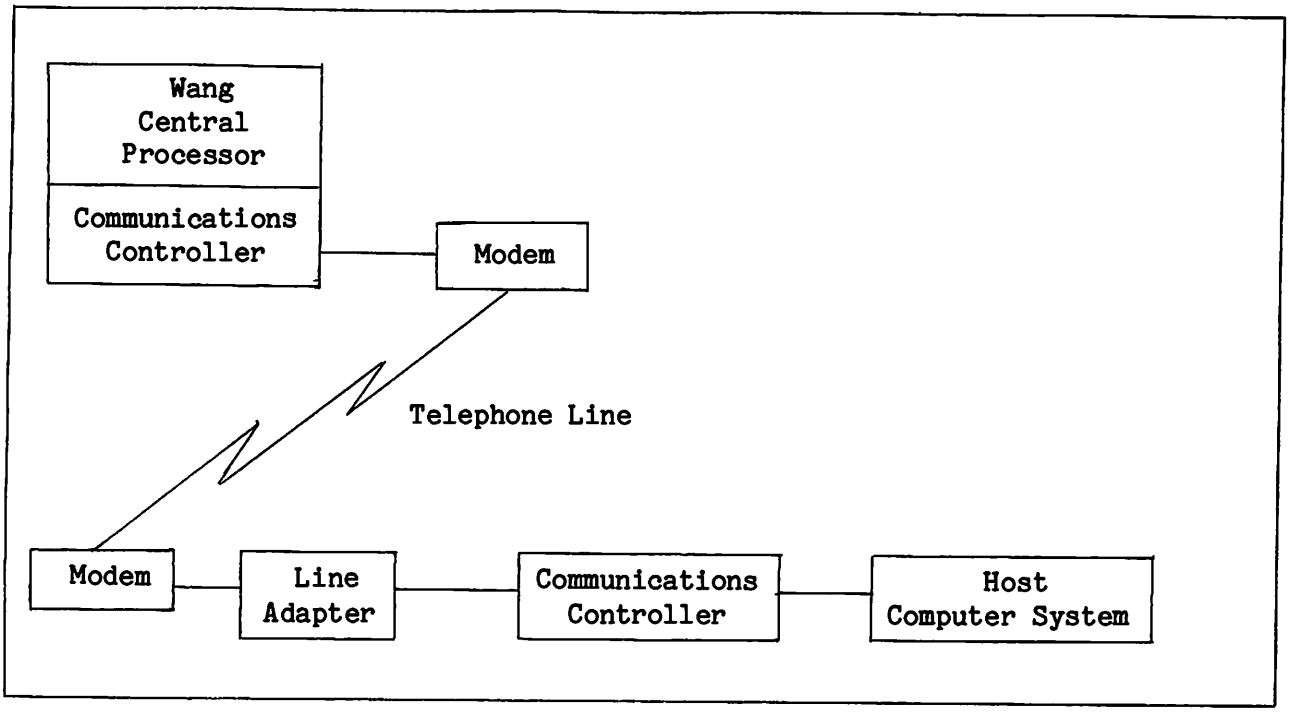


Figure 1-1. A Typical Remote Connection

The modem used with Wang's Synchronous/Asynchronous Communications Controller may be rented from the telephone company serving the locality where a Wang system is installed or may be purchased from any one of several modem vendors. However, in either case, arrangements with the telephone company may be necessary for connection of a modem to the telephone network.

Before a telephone company or a modem company representative arrives to install a modem, the location of the Wang system must be planned to ensure its proximity to the telephone equipment. Normally, modems are wired permanently to a wall; therefore, in such cases, subsequent relocation of the Wang system any great distance would necessitate relocation of the modem. EIA Standard RS-232-C recommends use of short cables (less than 50 feet or 15 meters) between data terminal equipment and communications equipment. Longer cable distances are possible only for operations at the lower range of transmission rates in an environment relatively free of electromagnetic interference.

Usually the compatible modems for a particular software package are listed in the documentation provided with the software. As one example, the compatible dial-up modems and supported line speeds for Wang's Binary Synchronous Communications (BSC) 2780, 3780, or 3741 emulation software are as follows:

Modem (Bell type or equivalent)	Line Speed
201A	2000 bits per second
201C	2400 bits per second
208B	4800 bits per second

As another example, the compatible modems for Wang's asynchronous Teletype or 2741 emulation software are as follows:

Modem (Bell type or equivalent)	Line Speed
103A or 103J	Up to 300 bps (full duplex)
212A	For 300 or 1200 bps (full duplex)
202C or 202S	Up to 1200 bps (half duplex)

Very likely, other RS-232-C compatible modems commonly used with asynchronous terminals having transmission rates in the range covered by the controller may prove suitable for user-developed asynchronous communications software. If acoustic couplers are used at both ends of a communications link, one acoustic coupler must have the "originate" feature and the other must have the "answer" feature, but ideally each acoustic coupler should have both features.

NOTE:

Remember two important requirements when ordering or attempting to use a modem:

1. A modem must be compatible with the software used in conjunction with the communications controller.
2. Modems used at both ends of a communications line must have the same characteristics (line speeds, asynchronous or synchronous type, etc.).

Do not expect to communicate with a remotely located host computer or other digital equipment if either of these requirements is not satisfied.

Direct connection of a Wang system with either a host computer system or another Wang system may be feasible under conditions such as the following:

- if the systems are in the same building,
- if electromagnetic interference is minimal,
- if the distance between the two systems does not exceed 50 feet (15.2m),
- if each system has a communications controller, and
- if extension cables and a null modem are obtained from Wang Laboratories.

Information about the extension cables and null modem is available from a Wang Sales or Service Office. (Also, see Appendix C.)

CHAPTER 2

SYNCHRONOUS COMMUNICATIONS

2.1 SOME GENERAL FEATURES

The "protocols" (the conventions and procedures) for synchronous type communications are inherently more complex than asynchronous protocols because information is transmitted in "blocks" rather than character-by-character. The blocks may consist of one or more records, and the records, in turn, may be fixed or variable in length. Special characters are used, as needed, to denote an end-of-record, an end-of-block, or an end-of-transmission; also, other special characters are used to denote a start-of-header and a start-of-text. Furthermore, to ensure data reliability, error checking techniques are incorporated on a block-by-block basis, and provision is made for retransmission of a data block upon request from a remote site.

When viewed as a group, synchronous transmission devices use a wide variety of block lengths. Often the physical nature of the data medium determines the block length. For example, if a device is transmitting punched card data, 80 characters per block is a convenient choice for the maximum block length. If a device is transmitting data from diskettes, 128 characters per block may be a convenient maximum block length. If a device is receiving data for output to a printer, the print line length may be the most convenient choice. In other cases, the size of the available buffers may become a limiting factor. Generally speaking, however, there are two conflicting trends associated with choosing a larger block size for a communications protocol:

1. Time is required between successive blocks for error checking, exchange of acknowledgment characters, and transmission of synchronization characters preceding a new block; thus, when fewer blocks (i.e., larger block sizes) are used for data transmission, the overall data transmission rate increases.

2. The probability of transmission errors increases with larger block sizes; each detected error produces a retransmission request, and the overall data transmission rate decreases.

Hence, the maximum block length may be a compromise value determined during system development and testing.

Among the many other concepts and special considerations associated with synchronous communications are the following:

- transparent and non-transparent transmission
- point-to-point and multipoint operations
- timeout functions (for transmit, receive, disconnect, and continue operations)
- record compression
- code translation.

Although discussions of these concepts go beyond the scope of this manual, the list should remind readers that a variety of features may be required when hardware and software are designed to support synchronous communications between terminals and a computer or between computer systems.

By utilizing loadable microcode to support synchronous communications applications, Wang's Synchronous/Asynchronous Communications Controller achieves flexibility and avoids obsolescence. As indicated in Section 1.1, microcode defining synchronous protocol tasks to be performed by the controller is produced only by Wang Laboratories and made available in terminal emulation software packages which also contain program modules defining input and output tasks to be performed by the central processor. The microcode is automatically loaded into the controller when the software is used.

Several of the most popular industry standard synchronous protocols (e.g., the binary synchronous 2780, 3780, 3741, and HASP multileaving protocols) are available in turnkey software. The software requires no changes in the host computer software for any mainframe computer which supports one or more industry standard protocols. Furthermore, Wang's software permits flexibility when choosing the peripherals to serve as input and output devices. For example, a card reader, diskette, or disk can serve as the input device in combination with a printer, diskette, or disk as the output device. The console keyboard is active for transmission of sign-on and other messages to the remote site, and the CRT is active for display of the prompts and error messages built into the software to ensure operational simplicity. Upon request, information is available from a local Wang Sales and Service Office or from Wang Laboratories, Inc.

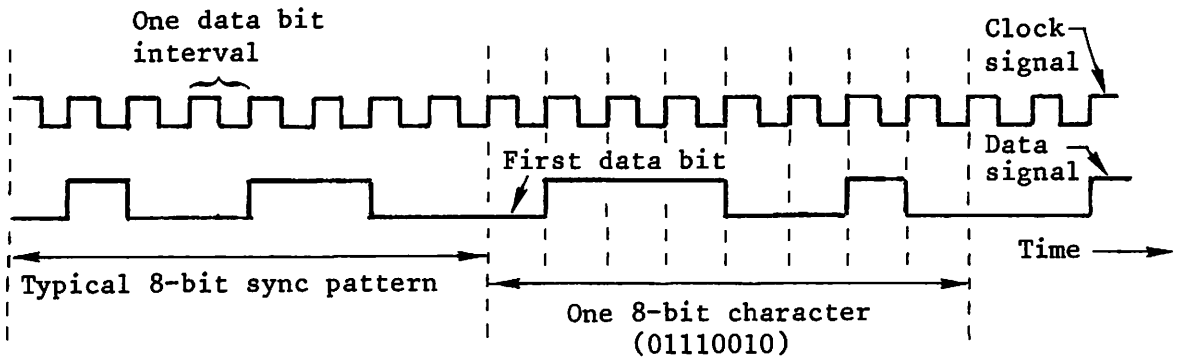
2.2 SYNCHRONOUS TRANSMISSION

As shown in Figure 2-1, synchronous data transmission requires the following:

1. A clock signal to mark the location of each data bit.

2. A synchronization pattern, added to the start of the transmission, to mark the first bit of the message.

3. Contiguous transmission of data bits following the synchronization pattern (since one data bit is assumed for every clock period).



Synchronous data transmission requires a clock signal and a synchronization pattern. Each data bit follows contiguously after the sync word, and one data bit is assumed for every clock period. A specified clock transition (rising or falling) marks the start of each data bit interval.

Figure 2-1. Synchronous Data Transmission

The clock signal requirement is one reason synchronous transmission equipment is more complex than asynchronous equipment which does not require a clock signal with the data signal. (See Figure 3-1.) On the other hand, synchronous transmission does not need synchronizing START/STOP elements for each character as is the case for asynchronous transmission equipment. The sync words preceding a synchronous transmission block represent less total communications channel overhead than the START and STOP elements in an asynchronously transmitted message.

CHAPTER 3

ASYNCHRONOUS COMMUNICATIONS

3.1 PROGRAMMABLE FEATURES

As indicated in Section 1.1, Wang's Synchronous/Asynchronous Communications Controller has built-in microcode to support asynchronous application programming. The microcode, residing in the controller's read only memory, implements the following standard and optional features when activated by an application program residing in the central processor:

- data buffering
- code translation
- substitution for characters received in error
- communications control, e.g., monitoring CPU ready-busy conditions, monitoring modem signals, implementing line turnaround procedures
- break signal detection and transmission
- detection of received timeouts
- detection of end-of-record characters
- automatic insertion and removal of shift characters
- sensing the Secondary Received Line Signal Detector and setting the Secondary Request to Send signal (also called reverse or supervisory channel data signals).

The controller's random access memory is used for purposes such as the following: (1) storage of initialization information, including code translation tables and a communications control vector, (2) storage of current status information, including parity and framing error detection for received data, binary counts of both the total number of characters and the number of end-of-record characters in the receive buffer, the received data timeout count down, and other conditions, and (3) storage of buffered input and output data. For each application, the desired transmission rate, communication mode, character format options, and some special operations are selected under program control when a $GIO statement in the application program loads the communications control vector information into the controller.

A selectable-speed clock on the controller supports serial asynchronous transmission and reception at line speeds from 50 to 9600 bps (bits per second). Any one of the following rates can be set via the initializing control vector: 50, 75, 100, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, or 9600 bps. Usually rates from 50 to 1800 bps are used for point-to-point, dial-up telecommunications; rates above 1800 bps are used for directly connected RS-232-C compatible equipment.

Any one of the following transmission modes is supported:

- half duplex (independent transmission one-way-at-a-time alternately)
- half duplex with automatic deletion of received null characters
- full duplex (independent transmission two-ways simultaneously)
- full duplex with automatic deletion of received null characters.

The desired mode is set via a particular byte in the ten-byte communications control vector.

Via other bytes in the vector, the asynchronous character format can be set as follows:

- odd, even, or no parity
- 5, 6, 7, or 8 data bits per character
- 1, 1.5, or 2 stop bits per character.

Figure 3-2 shows the byte-by-byte format of the communications control vector, and Table 3-1 lists the valid values representing the asynchronous transmission options. Also, Table 3-2 gives the format of the communications status vector, and Table 3-3 lists the $GIO microcommand sequences needed to operate the controller. An example in Section 3.14 illustrates some programming techniques.

3.2 ASYNCHRONOUS TRANSMISSION

The information in this section may be ignored by readers who are not interested in a detailed description of the character-by-character transmission procedure used by asynchronous equipment, in general, and Wang's communications controller, in particular.

Asynchronous transmission is often called start-stop transmission because each character is framed by start and stop elements as shown in Figure 3-1. The start element is represented by a transition from a logic "1" voltage level to a logic "0" voltage level. The nominal interval during which the logic "0" level is maintained for a start element is the same length as the interval used for each data bit. In some equipment, the data bit interval may be a fixed value or one of several possible values (if the transmission rate for the equipment is selectable). The length of the data bit interval is shorter when the transmission rate is higher.

Immediately following transmission of a start element, the voltage level is changed or not changed depending upon whether the first data bit is 1 or 0. (See Figure 3-1.) Similarly, after the first data bit interval, the voltage level is changed or not, as required, to represent the second data bit--and so on, successively, for each data bit. The number of transmitted data bits depends upon the equipment being used; it may be a fixed number for some equipment or a selectable number in other cases.

After the last data bit is transmitted, a parity bit may be transmitted if provisions for parity information are included in the equipment design. The parity bit interval is the same length as the data and start bit intervals. The voltage level may be a logic "0" or "1" depending upon the type of parity (odd or even) and the number of 1's occurring in the preceding data bits.

Finally, the stop element is transmitted using a logic "1" voltage level which is maintained until the next character is transmitted. Usually there is no upper limit to the length of a stop element; however, there is a lower limit depending upon the design of the transmitting equipment. The minimum length stop element may be a fixed value for some equipment or a selectable value in other cases.

Wang's Synchronous/Asynchronous Communications Controller provides the capability to communicate with a variety of asynchronous type equipment since there are selectable options for the character format, the parity, and the transmission rate.

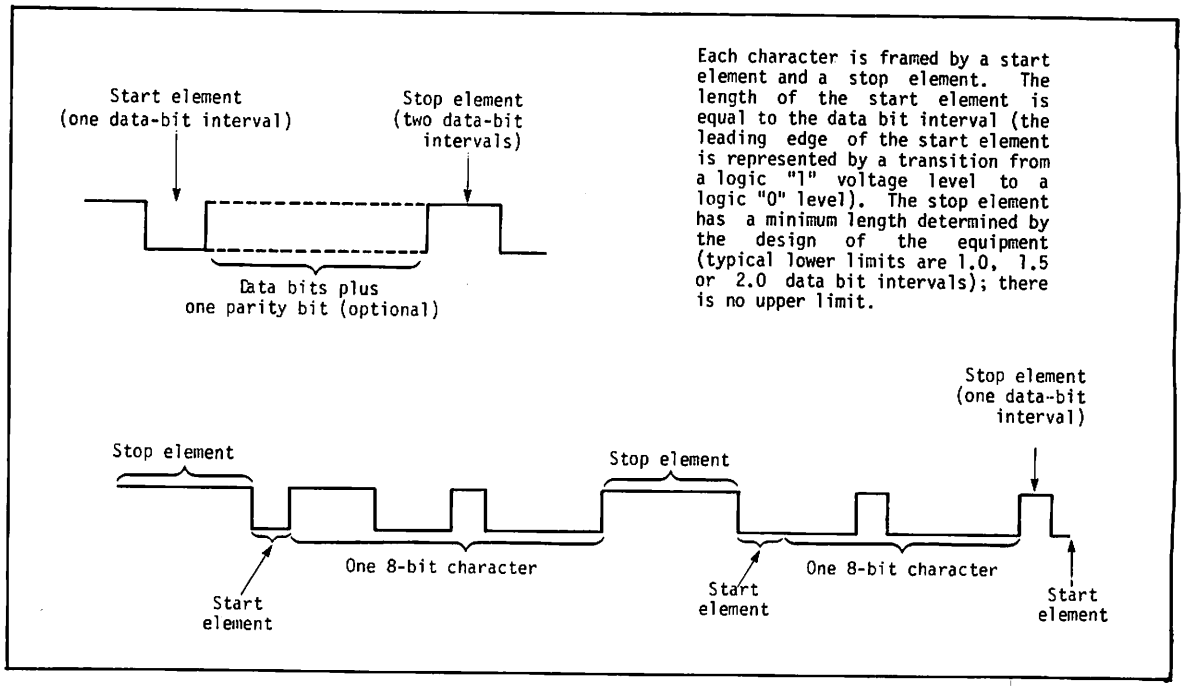


Figure 3-1. Asynchronous Data Transmission

Character Transmission

For asynchronous applications, Wang's communications controller transmits each character by modulating the transmitted data signal on Pin 2 in the connector as follows:

1. The transmitted data signal is set to "0" for one bit-time, representing the start bit.

2. Successively, low-order bit first, the signal is set for one bit-time to the value of each data bit until the number of transmitted data bits equals the number specified in the communications control vector; therefore, only the low-order 5, 6, 7, or 8 bits of a character are transmitted.

3. If odd or even parity is specified in the communications control vector, the signal is set for one bit-time to the appropriate value for the type of parity specified. In particular, if odd parity is specified, the parity bit is equal to 1 when the preceding data bits contain an even number of 1-bits; thus, for odd parity, the total number of 1's in the data bits plus the parity bit is an odd number. If even parity is specified, the parity bit is equal to 1 when the preceding data bits contain an odd number of 1-bits; thus, for even parity, the total number of 1's in the data bits plus the parity bit is an even number. If no parity is specified, step 3 is omitted.

4. The transmitted data signal is set to "1" for a minimum interval equal to 1, 1.5, or 2 bit-times, depending upon the number of stop bits specified in the communications control vector.

When no character is being transmitted, the transmitted data signal on Pin 2 is held at the value "1".

Character Reception

During asynchronous communications, the controller receives a character by detecting changes in the received data signal on Pin 3 in the connector as follows:

1. A transition from the voltage level representing logic "1" to the level representing logic "0" for at least one-half a bit-time is interpreted as the leading edge of the start bit for an incoming character.

2. The received data signal is sampled successively at times corresponding to the nominal center of each data bit. In particular, the nominal center of the first data bit is 1.5 bit-times after the leading edge of the start bit. The center of each subsequent bit occurs one bit-time after the center of its predecessor. Successively, low-order bit first, the bits in the character being received are set to correspond to the sampled values. The number of data bit samples taken by the controller equals the number of data bits specified in the communications

control vector. If the number of samples is less than 8, the remaining high-order bits in the received character are automatically set to 0 (unless the shift character option is in effect).

3. A parity bit, if specified, is read by sampling the received data signal again--one bit-time after the last data bit is sampled. The sampled parity value is compared with a calculated value based on the received data bits and the type of parity specified. If the received and calculated parity values are unequal, a designated bit in the communications status vector is set to 1 to indicate a parity error has occurred.

4. One bit-time after the received data signal is sampled for a parity value or the last data bit (if a "no parity" option is in effect), the signal is sampled again. Now, if the signal is "1", a valid stop bit is recognized. On the other hand, if the signal is "0", a framing error has occurred; consequently, a designated bit in the status vector is set to 1.

> NOTE:
>
> When developing or using an application program, remember that the transmission rate, the number of data bits, the type of parity, and the number of stop bits set for the communications controller must match the specifications for equipment in use at the other end of a communications link.

3.3 DATA BUFFERING

The controller has two multicharacter data buffers, a 175-byte transmit buffer and a 255-byte receive buffer. With these buffers, data transmission/reception operations performed by the controller with respect to the modem can overlap data input/output operations performed by the CPU with respect to the I/O peripherals designated for a communications application.

For example, after the CPU sends a data string to the controller, the CPU is free to perform an independent task such as fetching the next string of data to be transmitted from the input device--while the controller is performing such tasks as code translation, character formatting, and transmission to the modem.

> NOTE:
>
> If the transmit buffer becomes full while the CPU is sending data to the buffer, the data transfer rate from the CPU to the controller automatically slows to the rate at which characters are being transmitted from the buffer. No characters are lost.

On the other hand, the controller is free to receive a data string, perform operations such as code translation, and store the data in the receive buffer--while the CPU is performing an independent task such as outputting data to a designated peripheral.

> NOTE:
>
> If characters are received when the receive buffer is full, a buffer overrun condition occurs, and the appropriate error bit is set in the communications status vector. No other action is taken by the controller.

3.4 SUBSTITUTION FOR CHARACTERS RECEIVED IN ERROR

When a character is received with either a parity or a framing error, a substitute character (defined by byte 4 in the communications control vector) is automatically supplied by the controller; also, the appropriate error bit is set in the communications status vector. For example, a special character such as @ (or any other character not likely to occur in the incoming data) can be specified as the character to automatically replace any characters received in error. Replacement occurs before code translation, if any, is performed.

3.5 CODE TRANSLATION

The controller's code translation feature allows data interchange between the CPU and the controller in the ASCII code (American Standard Code for Information Interchange) native to Wang systems--regardless of the transmission/reception code for a particular application.

Space is reserved in the controller for two 256-byte code translation tables: (1) a transmit code translation table and (2) a receive code translation table. Specification of such tables is optional. Translation tables, supplied in the application program operating in the CPU, must be loaded into the controller by appropriate $GIO statements in the program. (See Section 3.12.)

The automatic code translation operation is enabled by loading a transmit or a receive code translation table (or both) after loading the communications control vector. If no tables are loaded, the code translation feature is effectively disabled.

During transmission, the code corresponding to a character sent from the CPU to the controller is used as an 8-bit index for a table lookup in the transmit code translation table. Then, an 8-bit character obtained from the table is placed in the transmit buffer. However, if byte 3 of the communications control vector specifies less than eight data bits per character, only the relevant low-order bits of each character are actually transmitted.

During reception, the code corresponding to a character received by the controller is used as an 8-bit index for a table lookup in the receive code translation table, and an 8-bit character is obtained from the table. If the translated character is a null character, $(00)_{16}$, and the high-order hexdigit in byte 2 in the communications control vector has the value 1 or 4, the character is discarded. Otherwise, the translated character is placed in the receive buffer.

> NOTE:
>
> Superfluous characters used for timing or fill can be removed automatically by translating them to null characters. This feature is applicable to half duplex and full duplex operations. (See Table 3-1.)

3.6 INSERTION AND REMOVAL OF SHIFT CHARACTERS

For applications involving data transmission and reception in a code set which utilizes shift characters (e.g., a Baudot code set or an IBM 2741 code set), a special feature of the communications controller removes the necessity for the program operating in the CPU to handle insertion and removal of shift characters. Instead, the upshift and downshift characters are defined in bytes 7 and 8 of the communications control vector. Also, the number of data bits per character is set to 5 or 6 (depending upon the application) by choosing an appropriate value for the high-order hexdigit in byte 3 of the communications control vector. Then, to activate automatic insertion and removal of shift characters, appropriate code translation tables are defined and loaded into the controller.

During data transmission, the controller examines and interprets the two high-order bits of each translated character in the transmit buffer as "shift status" bits as follows:

Two High-order Bits	Meaning
00	Downshifted character.
01	Upshifted character.
10	Character doesn't care about shift status.
11	Character doesn't care about shift status.

A shift character is automatically transmitted between any two characters whose shift status bits are different. In such cases, if the second character has upshifted status, an upshift character is transmitted prior to transmission of the second character. Alternatively, if the second character has downshifted status, a downshift character is transmitted prior to transmission of the second character. The shift status bits are not transmitted since the number of data bits actually transmitted per character is only 5 or 6 (low-order bits) if byte 3 in the communications control vector is appropriately specified.

During reception, the controller sets the value of the high-order bit of each received character (before code translation) according to the most recently received shift character as follows:

High-order Bit	Meaning
1	Upshifted character.
0	Downshifted character.

> NOTE:
>
> If a received character is a shift character, the character is discarded. Otherwise, the character is code translated and placed in the receive buffer.

3.7 DETECTING END-OF-RECORD CHARACTERS

The end-of-record detection feature is convenient for applications where a received data stream contains meaningful record delimiters, e.g., CR (carriage return) codes. The feature is particularly convenient for applications where there is no necessity to display the data while it is being received.

Since any number of characters can be defined as end-of-record characters, the controller can divide a received data stream into records and, thus, eliminate the need for the program operating in the CPU to perform the task.

To activate the end-of-record detection feature, byte 6 in the communications control vector must be set to HEX(01); also, a suitably defined receive code translation table must be loaded into the controller. To be suitably defined, the high-order bit for codes in the receive code translation table must be set to 1 for each character defined as an end-of-record character (and set to zero for all other characters).

During reception, if end-of-record detection is enabled, the controller maintains a count of the number of end-of-record characters currently stored in the receive buffer. This binary count is maintained in byte 5 of the communications status vector (see Table 3-2).

With an appropriate $GIO statement (see Section 3.12), the application program can read the status vector into the CPU. Then, the program can test the status information to ensure the availability of a complete record before requesting transfer of buffered data (via another $GIO statement whose microcommand sequence performs data transfer from the controller to the CPU).

When data is actually transferred from the controller's receive buffer to the CPU, only those characters up to (and including) the first end-of-record character are transferred. Furthermore, the high-order bit in the end-of-record character is changed from 1 to 0 when the character is transferred.

> NOTE:
>
> If the end-of-record detection feature is not needed for an application (or cannot be used because the high-order bit for codes in the receive code translation table is set to 1 for a purpose other than defining an end-of-record character), byte 6 in the communications control vector should be set to HEX(00) to disable end-of-record detection.

3.8 MONITORING RECEIVED TIMEOUTS

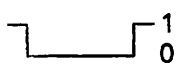
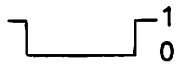
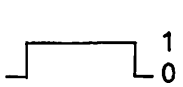
The communications controller has the capability to monitor received data timeouts. Byte 5 in the communications control vector is used to set the binary value of the timeout in units of 0.1 seconds. For example, the minimum timeout condition, 0.1 seconds, is specified by storing HEX(01) in byte 5. The maximum timeout condition, 25.5 seconds, is specified by storing HEX(FF) in byte 5. On the other hand, storing HEX(00) in byte 5 disables the monitoring feature for received timeouts.

> NOTE:
>
> If a timeout interval is specified, the controller maintains a received data timeout countdown in byte 6 of the communications status vector. (See Section 3.11.)

3.9 SENDING AND DETECTING BREAK SIGNALS

The communications controller has the capability to send and detect break signals under program control. Bytes 9 and 10 in the communications control vector are used to define the break signal transmission and detection intervals in units of 10 milliseconds. For example, HEX(14) stored in byte 9 defines an interval equal to 200 milliseconds for transmitted break signals. Similarly, HEX(11) stored in byte 10 defines an interval equal to 170 milliseconds for detection of break signals.

In addition to specifying the break signal intervals in bytes 9 and 10, it is necessary to use the low-order hexdigit position in byte 2 to specify the participating modem signals and the polarity of the break signals as follows:

Byte 2 (Low-order Hexdigit)	Break Signal Polarity	Modem Signals
0	none	none
1		Transmitted/Received Data
2		Secondary Request to Send, or Secondary Received Line Signal Detector
3		Secondary Request to Send, or Secondary Received Line Signal Detector

NOTES:

1. The Transmitted Data and Received Data modem signals are used with Bell 103 type modems.

2. Normally, the Secondary Request to Send and Secondary Received Line Signal Detector modem signals are used with Bell 202 type modems (which must be ordered with the reverse channel option in order to support break signal operation).

Transmission of a break signal by the controller involves inverting the level of the specified modem signal (i.e., Transmitted Data or Secondary Request to Send) for an interval defined by byte 9 of the communications control vector.

Detection of a break signal occurs when the controller senses the level of a specified modem signal (Received Data or Secondary Received Line Signal Detector) is continuously inverted for an interval at least as long as the interval defined by byte 10 of the communications control vector.

NOTE:

Detection of a break signal causes the "break signal received" bit in the status vector to be set. No other action is taken by the controller.

3.10 SPECIFYING THE COMMUNICATIONS CONTROL VECTOR

Figure 3-2 shows the format of the communications control vector, and Table 3-1 gives the valid specifications for the vector. The table is divided into two portions since the first three bytes of the vector are dual-purpose while the remaining bytes are single-purpose with respect to the available communications options and features.

In an application program, the control vector should be defined by a one-dimensional array having 10 elements with one byte per element. For example, to represent the control vector by the array C$(), use

```
DIM C$(10)1
```

Also, as a general programming practice, all elements should be initialized to binary zero by a statement of the form

```
INIT(00) C$()
```

before assigning values to particular elements in the array.

Then, as illustrated by the following statements, individual bytes in the control vector can be assigned values other than binary zero to select the desired options and define any special characters for the application.

Statement	Meaning
C$(1) = HEX(17)	One stop bit; 300 bits per second.
C$(2) = HEX(11)	Half duplex with automatic deletion of null characters; break enabled on transmit/receive.
C$(3) = HEX(23)	Seven data bits; odd parity.
C$(4) = HEX(5E)	Substitute character for parity or framing error is an up-arrow, ↑ .
C$(5) = HEX(0A)	Timeout interval is 1 second.

When defining some special characters, remember to choose a compatible value in the first three bytes of the communications control vector. For example, if defining upshift and downshift characters in bytes 7 and 8, the high-order hexdigit in byte 3 must have the value 0 or 1 (since the shift feature can be used only with code sets having 5 or 6 data bits per character). See Section 3.6 and Table 3-1.

> NOTE:
>
> The communications control vector must be loaded into the controller via a $GIO statement containing the appropriate microcommand sequence from Table 3-3. (See Section 3.12.)

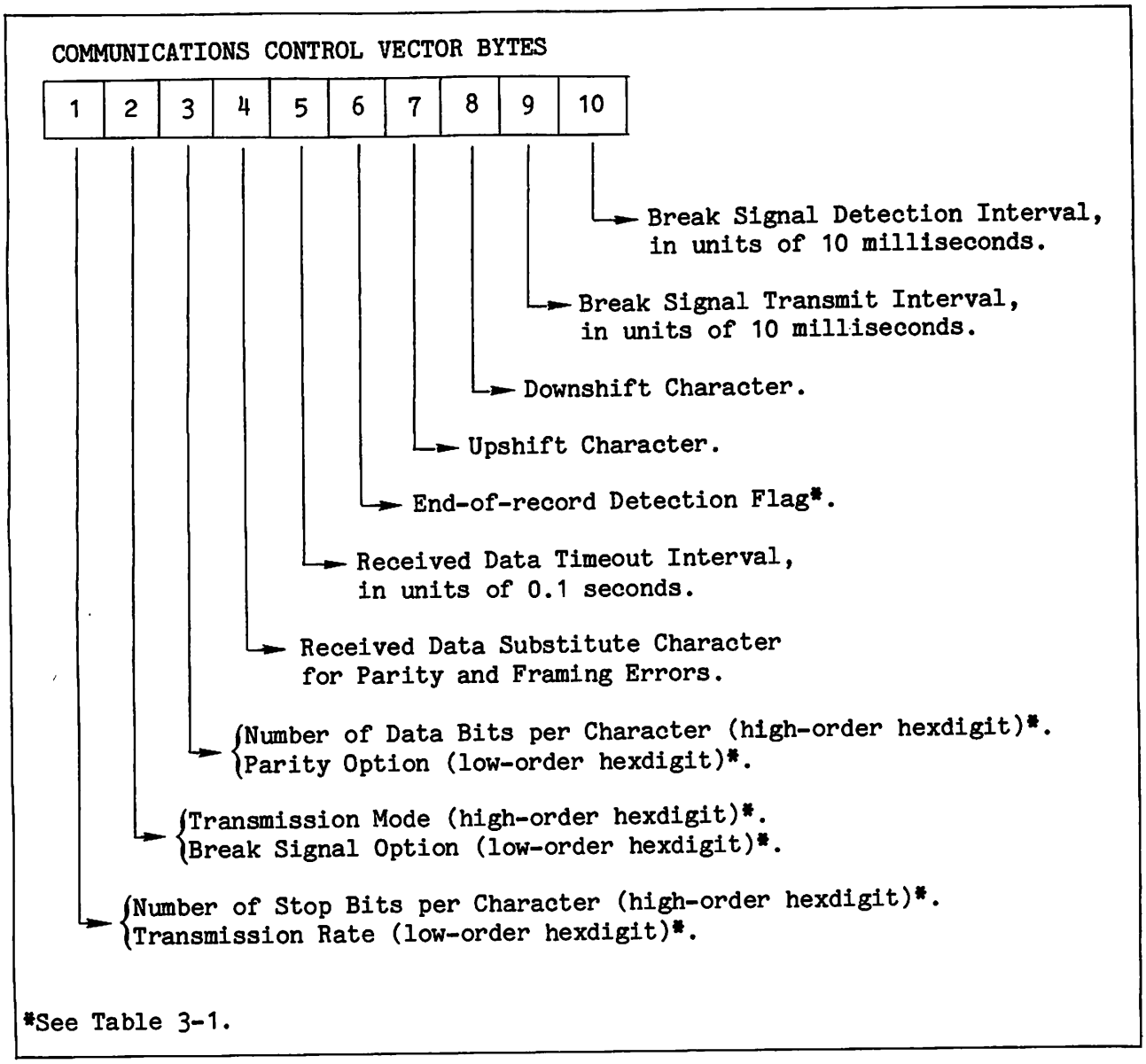


Figure 3-2. Communications Control Vector Format

Table 3-1. Valid Communications Control Vector Specifications

Byte*	High-order Hexdigit	Low-order Hexdigit
1	0 = Illegal value 1 = 1 Stop bit 2 = 1.5 Stop bits 3 = 2 Stop bits	0 = 50 bps (bits per second) 1 = 75 bps 2 = 100 bps 3 = 110 bps 4 = 134.5 bps 5 = 150 bps 6 = 200 bps 7 = 300 bps 8 = 600 bps 9 = 1200 bps A = 1800 bps B = 2400 bps C = 3600 bps D = 4800 bps E = 7200 bps F = 9600 bps
2	0 = Half duplex 1 = Half duplex with deletion of received null characters 2 = Full duplex 3 = Full duplex with deletion of received null characters	0 = Break disabled 1 = Break enabled on transmit/receive 2 = Break enabled on Secondary Req. to Send & Sec. Rec. Line Sig. Det. 3 = Same as 2 with inverted polarity
3	0 = 5 Data bits per character 1 = 6 Data bits 2 = 7 Data bits 3 = 8 Data bits	0 = No parity 1 = Even parity 2 = No parity 3 = Odd parity

*For bytes 4 through 20, see Table 3-1 (Continued).

Table 3-1. Valid Communications Control Vector Specifications (Continued)

Byte	Hexadecimal Notation*	Remarks
4	xy = Substitute character for parity/framing errors.	Each received character having a parity or framing error is replaced by the designated character (replacement occurs prior to code translation if translation tables are being used). See Section 3.4.
5	xy = Timeout interval in units of 0.1 seconds.	The specification in hexadecimal notation represents the timeout interval in units of 0.1 seconds, e.g., $(24)_{16} = (36)_{10}$ specifies an interval of 3.6 seconds. See Section 3.8.
6	00 = Disable end-of-record detection. 01 = Enable end-of-record detection.	If enabled, the end-of-record characters must be defined via the receive code translation table by setting the high-order bit to 1 for each code corresponding to an incoming end-of-record character. See Section 3.7.
7	xy = Upshift character.	To enable shift code insertion/deletion, the high-order hexdigit in byte 3 of the control vector must be 0 or 1 (i.e., the number of data bits per character must be 5 or 6). Also, the transmit code translation table must identify all downshifted, upshifted, and "don't care" characters by setting the two high-order bits to 00, 01, and either 10 or 11 as described in Section 3.6. The receive code translation table must allow for the controller's automatic setting (before translation) of the high-order bit to 1 for all incoming upshifted characters.
8	xy = Downshift character.	
9	xy = Break signal transmit interval in units of 10 ms.	To enable break signal transmission/detection, the low-order hexdigit in byte 2 of the control vector must specify the polarity and the modem signals. If bytes 9 and 10 are both HEX(00), the low-order hexdigit in byte 2 should be 0. The byte 9 and 10 specifications in hexadecimal notation represent break signal transmit and receive intervals in units of 10 milliseconds, e.g., $(12)_{16} = (18)_{10}$ specifies a 180 ms interval. See Section 3.9.
10	xy = Break signal detection interval in units of 10 ms.	

*x and y each denote any hexdigit (0 through 9, A through F). If a feature is not desired, the byte positions associated with the feature can be ignored if the communications control vector has been initiated to binary zero.

3.11 THE COMMUNICATIONS STATUS VECTOR

Space is reserved in the controller's random access memory for a communications status vector whose byte and bit positions are used automatically as shown in Table 3-2. The first three bytes of the status vector are cleared automatically when the communications control vector is loaded into the controller from the CPU, and whenever the status vector is read by the application program. (See Section 3.12.)

Several flags are set in particular bit positions in the first three bytes of the status vector during controller operations. Also, binary counts for the current number of characters in the receive buffer and the number of end-of-record characters are maintained in bytes 4 and 5. Similarly, a binary count for the current number of characters in the transmit buffer is maintained in byte 7. Byte 6, on the other hand, is similar to a real time clock whose value is initiated to the received data timeout interval (as specified in byte 5 of the communications control vector) each time one of the following events occurs:

a) a $GIO "start receiving data" operation begins,

b) a line turnaround occurs during a $GIO "send, then receive data" operation, or

c) a character is received during either operation.

However, if the byte 6 value is not reset by one of these operations, the countdown proceeds to zero.

Whenever desired, the status vector information can be read (transferred to the CPU) by a $GIO statement containing the appropriate microcommand sequence from Table 3-3. (See Section 3.12.) After transfer to the CPU, status vector information can be tested, as required, by the application program.

Table 3-2. Communications Status Vector Information

Byte	Bit*	Meaning
1	1	1 = Break signal received.
2	1 2 3	1 = Received Line Signal Detector On. 1 = Sec. Rec'd Line Sig. Det. On. 1 = Data Set Ready modem signal On.
3	1 2 3	1 = Receive parity error detected. 1 = Receive buffer overrun error detected. 1 = Receive framing error detected.
4	all	Binary count of the number of characters in the receive buffer.
5	all	Binary count of the number of end-of-record characters in the receive buffer.
6	all	Received data timeout countdown.
7	all	Binary count of the number of characters in the transmit buffer.

*Bit positions in each byte are numbered from 1 (low-order) to 8 (high-order).

3.12 CPU AND CONTROLLER INTERACTION VIA $GIO STATEMENTS

To operate the Synchronous/Asynchronous Communications Controller for an asynchronous application, a user-developed application program residing in the CPU should include $GIO statements with suitable microcommand sequences. A list of valid microcommand sequences for controller operations is presented in Table 3-3.

Table 3-3. Microcommand Sequences for Controller and CPU Interaction

Controller and CPU Interaction	Microcommand Sequence*	Remarks
Set communications control vector	4402 A000 440C	
Read communications status vector	(See ** below)	
Load transmit code translation table	4404 A000 440C	
Load receive code translation table	4405 A000 440C	
Disconnect	4406	
Send break signal	4407	
Start receiving data	4408	For half or full duplex.
Transfer received data to CPU	(See ** below)	For half or full duplex.
Send data	440A A000 440C	For half or full duplex.
Send, then receive data	440B A000 440C	For half duplex only.
Stop transmitting	440C	For full duplex protocols.
Continue transmitting	440D	For full duplex protocols.

*A microcommand sequence can be specified directly or indirectly in a $GIO statement. If the microcommand sequence is specified directly (as the arg-1 component), each four-hexdigit-code can be separated from the previous one by a space for readability as shown in this table. If the microcommand is specified indirectly (by assigning the sequence to a variable and using the variable as the arg-1 component), spaces cannot be used between the four-hexdigit-codes, e.g., A$ = HEX(4402A000440C). Furthermore, the dimension of the variable must be large enough to ensure the presence of two trailing space characters which serve as the pseudo-microcommand 2020 denoting the end of the sequence. Unpredictable results may occur if at least one trailing blank does not follow an indirectly specified microcommand sequence. (See the discussion of $GIO in the BASIC or BASIC-2 language reference manual provided with the Wang System.)

**The valid microcommand sequence is dependent upon whether the controller is installed in either a 2200MVP central processor or some other 2200 central processor. See the following table.

Controller and CPU Interaction	non-2200MVP	Sequence for 2200MVP
Read communications status vector	4403 C620	4403 1020 02FF 03FF 1223 C620
Transfer received data to CPU	4409 C620	4409 1020 02FF 03FF 1223 C620

Brief descriptions of the operations in Table 3-3 follow. A sample $GIO statement is shown for each operation; however, in a user-developed program, different comments and variables may be used in the $GIO statements. Also, the address 01C (which corresponds to the standard peripheral address for the controller) may not be appropriate for the communications controller in the system being used.

Set Communications Control Vector

```
$GIO SET CCV /01C (4402 A000 440C, G$) C$()
```

The communications control vector, defined here by the array C$(), is set (loaded into the controller) when the statement is executed. Here, 01C is the address of the controller, and G$ represents the error/status/general-purpose registers.

> NOTE:
>
> The controller's transmit and receive buffers, as well as the communications status vector and code translation tables, are cleared automatically when the communications control vector is loaded.

Read Communications Status Vector

```
$GIO READ CSV /01C (4403 C620, G$) A$
```

or for a 2200MVP central processor

```
$GIO READ CSV /01C (4403 1020 02FF 03FF 1223 C620, G$) A$
```

The information currently in the communications status vector is read into the CPU and stored in the character string A$ (which must be at least 7 bytes long).

> NOTE:
>
> The error and received break indicators (i.e., bytes 1 and 3) in the communications status vector are cleared automatically after the status vector information is read into the CPU. In a program for a 2200MVP central processor, insert a line containing a $BREAK statement before each line containing a "read communications status vector" statement.

Load Transmit Code Translation Table

```
$GIO LOAD TTBL /01C (4404 A000 440C, G$) C1$()
```

The transmit code translation table is loaded into the controller from the array C1$() if such an array is previously defined in the application program. The optional transmit code translation feature is enabled only if a transmit code translation table is loaded after the communications control vector is loaded into the controller.

NOTES:

1. The transmit code translation table should be exactly 256 bytes long. The byte positions in the table should contain the non-ASCII character codes (the "after translation characters") arranged in a sequence corresponding to Wang's ASCII character codes (the "before translation characters"). In effect, the translation procedure uses the binary equivalent of an outgoing character's hexadecimal code as an index for a table lookup operation by which the appropriate translation character is found. For example, if the outgoing ASCII character is an uppercase B, i.e., HEX(42), whose binary value is 66, the corresponding non-ASCII character should be located in the 67th position of the transmit code translation table. (Remember that the first position in the table corresponds to the binary value zero.)

2. If shift character automatic insertion/removal is in effect (i.e., the specified number of data bits per character is 5 or 6), the two high-order bits of each code in the transmit code translation table must conform to the appropriate values given in Section 3.6.

Load Receive Code Translation Table

```
$GIO LOAD RTBL /01C (4405 A000 440C, G$) C2$()
```

The receive code translation table is loaded into the controller from the array C2$() if such an array is previously defined in the application program. The optional receive code translation feature is enabled only if a receive code translation table is loaded after the communications control vector is loaded into the controller.

NOTES:

1. The receive code translation table should be exactly 256 bytes long. The byte positions in the table should contain ASCII character codes (the "after translation characters") arranged in a sequence corresponding to the non-ASCII character codes (the "before translation characters"). In effect, the translation procedure uses the binary equivalent of an incoming character's hexadecimal code as an index for a table lookup operation by which the appropriate translation character is found. For example, if the incoming non-ASCII character is a HEX(18), the binary value is 24; therefore, the corresponding ASCII character should be located in the 25th position of the receive code translation table. (Remember that the first position in the table corresponds to the binary value zero.)

2. If shift character automatic insertion/removal is in effect, the 256-byte receive code translation table represents two 128-byte tables. The first 128 bytes in the table should represent the conversions for incoming downshifted characters corresponding to the hexadecimal codes HEX(00) through HEX(7F). The second 128 bytes should represent conversions for incoming upshifted characters corresponding to the hexadecimal codes HEX(80) through HEX(FF).

3. If end-of-record character detection is enabled, the high-order bit for codes in the receive code translation table must be set to 1 for each character defined as an end-of-record character (and set to zero for other characters).

Disconnect

```
$GIO DISCONNECT /01C (4406, G$)
```

The controller disconnects from the line by setting the Data Terminal Ready signal to zero for a period of three seconds.

Send Break

```
$GIO BREAK /01C (4407, G$)
```

The controller sends a break signal in accordance with the circuit and polarity denoted by the low-order hexdigit in byte 2 of the communications control vector. See Table 3-1 and Section 3.9.

Start Receiving Data

```
$GIO START RCV /01C (4408, G$)
```

One "start receiving data" statement is needed to enable data reception via the controller, whether set for the full or half duplex mode. If set for half duplex mode, the transmit and receive buffers are cleared first. The controller enters the receive mode and starts receiving data. Also, the receive timeout countdown is started by initiating byte 6 of the communications status vector to the value specified as the timeout interval (if different from binary zero).

Transfer Received Data to the CPU

```
$GIO RCV /01C (4409 C620, G$) D$()
```

or for a 2200MVP central processor

```
$GIO RCV /01C (4409 1020 02FF 03FF 1223 C620, G$) D$()
```

All or part (if an end-of-record character is detected) of the receive buffer characters are transferred from the controller to the CPU and stored in the array D$(). (See Section 3.7.) The $GIO data buffer, denoted here by D$(), should be at least 255 bytes long since the controller has a 255-byte receive buffer. Bytes 9 and 10 in the error/status/general-purpose registers provided by the variable G$ (i.e., arg-2 of the $GIO statement) are set to the binary representation of the number of bytes transferred, whether stored or not. In a program for a 2200MVP central processor, insert a line containing a $BREAK statement before each line containing a "transfer received data" statement.

Send Data

```
$GIO SEND /01C (440A A000 440C, G$) F$() <1, N>
```

If set for half duplex mode, the receive buffer is cleared first. For half or full duplex mode, bytes 1 through N of the array F$() are transferred from the

CPU to the controller where they are stored in the transmit buffer after code translation is performed, if enabled. Then, the controller transmits the data and remains in the transmit mode (if set for the half duplex operation).

NOTES:

1. The $GIO microcommand A000 implements a particular signal sequence repeatedly (once per character until each character in the arg-3 data buffer is transferred from the CPU to the controller). In Wang BASIC (but not BASIC-2), the $GIO syntax requires a single argument format for arg-3. Therefore, generally speaking, a $PACK statement should be used to pack multiargument data into a single argument prior to executing the "send data" $GIO statement--if the application requires a specially formatted buffer.

2. If desired, data can be transmitted via the controller using a PRINT, PRINTUSING, or MAT PRINT statement in conjunction with $GIO statements by employing techniques such as those described in Section 3.13.

Send Then Receive Data

```
$GIO SEND RCV /01C (440B A000 440C, G$) F$()  <E>
```

This statement is applicable only for the half duplex mode of operation. Beginning with the Eth byte, all remaining bytes of the array F$() are transferred from the CPU to the controller for storage in the transmit buffer after code translation is performed, if enabled. The controller transmits the data and then executes a "start receiving data" operation.

NOTES:

1. For half duplex communications, the "send data" $GIO operation should be used to send all but the last bytes of data. The "send, then receive data" $GIO operation should be used to send the last bytes of data. Afterwards, the "transfer received data to CPU" $GIO operation should be used. (The "send, then receive data" $GIO operation automatically implements a line turnaround procedure, thereby ensuring the controller's readiness to receive data without loss.)

2. For full duplex communications, only the "send data" and "transfer received data to CPU" $GIO operations are needed. The "send, then receive data" $GIO operation should not be used since the controller (in full duplex mode) remains in both the transmit and receive modes simultaneously and line turnaround does not occur.

Stop Transmitting

```
$GIO STOP SEND /01C (440C, G$)
```

This statement is applicable for the full duplex mode of operation, in cases where the CPU must stop transmission temporarily because a control sequence is received without clearing the contents of the transmit buffer. Transmission commences when a "send data" $GIO operation or a "continue transmitting" $GIO operation is executed.

Continue Transmitting

```
$GIO CONTINUE SEND /01C (440D, G$)
```

This statement can be used to restart transmission if the transmit buffer contains data and transmission has been halted by a "stop transmitting" $GIO operation.

3.13 TRANSMISSION VIA PRINT, PRINTUSING, OR MAT PRINT STATEMENTS

If desired, data transmission can be implemented via PRINT, PRINTUSING, or MAT PRINT statements by employing the special technique described here. The technique effectively preserves the structure of the "send data" $GIO operation (described in Section 3.12) by breaking the operation into three phases, two of which must be replaced by $GIO operations introduced in this section.

Consider the following statement:

```
100 $GIO SEND DATA /01C (440A A000 440C, G$) A$()
```

and note the three phases corresponding to the microcommand sequence--

- 440A — Sends the code $(0A)_{16}$ to the controller via a CBS strobe to initiate data transmission.
- A000 — Performs data transfer using a prescribed sequence repeatedly (once for each character) until all data in the arg-3 buffer A$() is transferred from the CPU to the controller.
- 440C — Sends $(0C)_{16}$ to the controller via a CBS strobe to terminate data transmission.

Now, for convenience, let's define two operations not shown in Table 3-3:

Operation	Microcommand Sequence
Start send	440A
End send	440C

Using the two new $GIO operations, consider the following alternative programming sequence as a replacement for the "send data" $GIO operation shown in the previous line 100:

```
100 $GIO START SEND /01C (440A, G$)
110 SELECT PRINT 01C (185)
120 PRINT X(7); B$(3); C$(5,12)
130 SELECT PRINT 005 (64)
140 $GIO END SEND /01C (440C, G$)
```

In the sequence represented by lines 100 through 140, the PRINT statement in line 120 effectively replaces the microcommand A000 (the data transfer phase) in the original line 100. A PRINTUSING or a MAT PRINT statement could be used in line 120.

The technique described in this section must not occur in the programming logic until after the communications control vector has been set. (See Section 3.10 and Table 3-1.)

3.14 AN EXAMPLE

The program listed in this section illustrates how a Wang system equipped with a Synchronous/Asynchronous Communications Controller can be programmed to emulate a Teletype terminal. The Wang keyboard corresponds to the Teletype keyboard, and the CRT corresponds to the Teletype printer. Many REM statements are included in the program to highlight special features such as the following:

1. An asynchronous format with 7 data bits per character, even parity, and 1 stop bit is specified.

2. Rate = 300 baud (i.e., line speed is 300 bits per second).

3. Mode = half duplex with automatic deletion of null characters.

4. Break signal transmission/detection is enabled with a 200 ms transmission interval and a 120 ms detection interval.

5. End-of-record detection is enabled. The carriage-return and DC1 (X-ON) characters are defined as terminators in the receive code translation table. Each carriage-return $(0D)_{16}$ is translated to $(8D)_{16}$ prior to storage in the receive buffer. Each DC1 or X-ON character is translated to $(A0)_{16}$ prior to storage in the receive buffer. Upon transfer to the CPU, the high-order bit of these end-of-record characters is changed from 1 to 0; hence, each $(8D)_{16}$ becomes $(0D)_{16}$ which is the ASCII code for a carriage-return, and $(A0)_{16}$ becomes $(20)_{16}$ which is the ASCII code for a space character.

NOTE:

The choice of translation characters is somewhat arbitrary and actually depends upon whether a programmer wishes to preserve the identity of an incoming end-of-record character. To preserve the identity, change the high-order bit of the incoming code to 1 to obtain the proper translation code (e.g., the incoming hex code 0D is changed to 8D in the program which follows). On the other hand, if an incoming character is not suitable for subsequent printing or processing, first choose a desired replacement character and then change the high-order bit of the replacement code to 1 to obtain the proper translation code (e.g., if a space character is chosen as the desired replacement character, its hex code 20 becomes A0 when the high-order bit is changed to 1).

6. Also, in the sample program, characters received with a parity or framing error are replaced by the substitute character $(7F)_{16}$, the ASCII code for a DEL character. Via the receive code translation table, each $(7F)_{16}$ is converted to a null character, i.e., $(00)_{16}$.

If desired, the program can be keyed into the CPU and saved on a disk or diskette to serve as a test program. However, remember that the program incorporates special features and cannot be used unless the following conditions are satisfied:

1. If the address of the controller is not 01C, change the SELECT statement in the program accordingly. See line 110.

2. A suitable modem must be available, and modems at both ends of the communications link must be similar. See Section 1.3.

3. The number of data bits per character, the number of stop bits, the type of parity, and the transmission rate must be matched at both ends of the communications link. If necessary, adjust the values in the communications control vector. See lines 300 through 390.

4. If attempting to communicate with a host computer, find out what sign-on procedure is required.

The Sample Program (Requires modification for 2200MVP central processors)

```
10 REM EXAMPLE --TTY EMULATION-- KYBD FOR INPUT, CRT FOR OUTPUT
20  DIM C2$(16)16, L$(255)1,  K$1, X$(10)1, Z$(7)1
21 REM C2$() IS A 256-BYTE RECEIVE CODE TRANSLATION TABLE
22 REM L$() IS A 255-BYTE CPU RECEIVE DATA BUFFER
23 REM K$ IS A 1-BYTE CPU KEYBOARD INPUT BUFFER
24 REM X$() IS A 10-BYTE COMMUNICATIONS CONTROL VECTOR
25 REM Z$() IS A 7-BYTE CPU ARRAY FOR READING STATUS VECTOR
30 REM .....INITIALIZATION MODULE BEGINS
40 REM ..DEFINE $GIO MICROCOMMANDS TO OPERATE CONTROLLER
50   G0$=HEX(4402A000440C)          :REM  SET CONTROL VECTOR
60   G1$=HEX(4403C620)              :REM  READ STATUS VECTOR
70   G3$=HEX(4405A000440C)          :REM  LOAD RCV TRANSLATE TABLE
80   G6$=HEX(4408)                  :REM  START RECEIVING DATA
90   G7$=HEX(4409C620)              :REM  TRANSFER RECEIVED DATA
100  G9$=HEX(440BA000440C)          :REM  SEND-THEN-RECEIVE DATA
110  SELECT #1 01C                  :REM  SELECT 2227B AS #1
120 REM ..DEFINE RCV TRANSLATION TABLE
130  INIT(00)C2$()              :REM  CLEAR RCV TRANSLATION TABLE
140  C2$(1)=HEX(000102030405060708090A0B0C0D0E0F)   :REM  00-0F
150  C2$(2)=HEX(101A1213141516171819A1B1C1D1E1F)   :REM  10-1F
160  C2$(3)=HEX(202122232425262728292A2B2C2D2E2F)   :REM  20-2F
170  C2$(4)=HEX(303132333435363738393A3B3C3D3E3F)   :REM  30-3F
180  C2$(5)=HEX(404142434445464748494A4B4C4D4E4F)   :REM  40-4F
190  C2$(6)=HEX(505152535455565758595A5B5C5D5E5F)   :REM  50-5F
200  C2$(7)=HEX(606162636465666768696A6B6C6D6E6F)   :REM  60-6F
210  C2$(8)=HEX(707172737475767778797A7B7C7D7E00)   :REM  70-7F
220 REM ..SPECIAL MEANINGS IN THE ABOVE TABLE ARE AS FOLLOWS
230 REM HEX 0D  (CARRIAGE RETURN) SET AS TERMINATOR
240 REM HEX 11  (DC1, X-ON) SET AS TERMINATOR/SHOW AS SPACE
250 REM HEX 7F  (DEL, RUBOUT) CONVERT TO NULL
255 REM HEX 80 THRU FF CONVERT TO NULL
260 REM ..DEFINE COMMUNICATIONS CONTROL VECTOR
280  INIT(00)X$()           :REM  INITIALIZE CCV TO BINARY ZERO
290 REM ..IF THE CCV IS NOT SET TO 00, TIME DELAYS MAY OCCUR.
300  X$(1)=HEX(17)          :REM  STOP BITS=1, BAUD RATE=300
310  X$(2)=HEX(11)          :REM  MODE=HALF-DUPLEX, BREAK ENABLED
320  X$(3)=HEX(21)          :REM  DATA BITS=7, PARITY=EVEN
330  X$(4)=HEX(7F)          :REM  ERROR SUBSTITUTE CHARACTER=DEL
350  X$(6)=HEX(01)          :REM  END OF RECORD DETECTION
380  X$(9)=HEX(14)          :REM  BREAK SEND INTERVAL=200 MS
390  X$(10)=HEX(0C)         :REM  BREAK DETECT INTERVAL=120 MS
400 REM ..
410  PRINT HEX(03),,"EMULATE TTY 300 BAUD"  :REM CLEAR CRT
415 PRINT TAB(9);"KEYBOARD FOR INPUT--CRT FOR OUTPUT"
420  D, I=1:PRINT
430  $GIO SET CONTROLS #1 (G0$,A$)X$()
440  $GIO SET RCV TABLE #1(G3$,A$)C2$()
450  $GIO START RCV #1 (G6$,A$)
455 REM .....END OF INITIALIZATION MODULE
```

(continued on next page)

```
456 REM ..PROMPT OPERATOR
460  PRINT "**********BEGIN SIGN-ON PROCEDURE**********"
470  PRINT TAB(5);"NOTE---S.F. '15 IS PROGRAMMED TO SEND A BREAK
 SIGNAL."
480  GOTO 540
500 REM ....MAIN LOOP BEGINS
510 REM ..OUTPUT KEYED DATA TO 2227B
520  $GIO SEND DATA #1(G9$,A$)K$
530 REM ..KEYBOARD/T.C. TEST LOOP
540  $IF ON /001,770                          :REM TEST KYBD READY
550  $GIO READ STATUS #1 (G1$,A$)Z$()
560  IF Z$(1)>HEX(00)THEN 740                 :REM  TEST FOR BREAK
570  IF Z$(3)>HEX(00)THEN 730                 :REM  TEST FOR ERRORS
580  IF Z$(4)=HEX(00)THEN 540                 :REM  TEST FOR COUNT ZERO
590  $GIO TRANSFER RCVD DATA #1 (G7$,A$) L$()<I>
600  A= VAL(STR(A$,10))                       :REM  A IS COUNT
610  IF A+D>64 THEN 650                       :REM  BRANCH IF A>=64
620  $GIO /005(A000,A$)L$()<I,A>              :REM  DATA TO CRT IF A<64
630  D=D+A  :GOTO 690                         :REM  D IS OUTPUT POINTER
640 REM ..DISPLAY OVERRUN
650  B=65-D                                   :REM  B IS LINE LENGTH
660  $GIO/005(A000 400D 400A,A$)L$()<I,B>:REM DATA,CR,LF TO CRT
670  D=A-B
680  $GIO /005(A000,A$)L$()<I+B,D>            :REM  NEXT LINE TO CRT
690  I=I+A  :A=I-1
700  IF L$(A)<>HEX(0D)THEN 540
710  PRINT
720  INIT(00)L$() :D,I=1   :GOTO 540
725 REM ..L$() IS CLEARED WHEN A C.R. IS RECEIVED
730  GOTO 550                                 :REM  RCV ERROR DETECTED
740  PRINT "...BREAK RECEIVED":GOTO 540
750 REM ...KEYBOARD LOGIC FOLLOWS
770  SELECT PRINT 005:KEYIN K$,790,880  :REM  ACCEPT KYBD INPUT
790  IF K$<HEX(20) THEN 830
800  PRINT K$;                                :REM  DISPLAY K$ ON CRT
810  D=D+1:IF D<65 THEN 520:PRINT :D=1:GOTO 520
820  PRINT :INIT(00)L$():D,I=1:GOTO 520
825 REM ..BRANCH AS FOLLOWS FOR A CODE HEX 08 THRU 0D
830  ON VAL(K$)-7 GOTO 850,800,840,520,520,820  :GOTO 520
840  PRINT K$;:GOTO 520
850  D=D-1:IF D>0 THEN 860:D=64:PRINT HEX(0C);
860  PRINT HEX(082008);:GOTO 520
870 REM ..BRANCH AS FOLLOWS FOR AN S.F. CODE HEX 07 THRU 0F
880  ON VAL(K$)-6 GOTO 840,850,800,840,520,520,520,520,890   :GO
TO 520
890  PRINT "....SEND BREAK":$GIO #1(4407,A$)
900  GOTO 540
910 REM ...END OF KEYBOARD LOGIC
920 REM ....END OF MAIN LOOP
```

(See notes on next page.)

NOTES:

1. Wang's Teletype emulation software has many features not illustrated in the example given in this section. With the software, data transmission and reception can be controlled over point-to-point, dial-up communications links between Wang systems and host computer systems which support Teletype-like line protocols. From the viewpoint of a Wang system operating under control of a Teletype emulation program, the keyboard is always active as an input device for data transmission, and the CRT is always active as an output device for data reception. Additionally, and optionally, stored data can be transmitted from a disk or diskette, and received data can be output to a printer, disk, or diskette. The active I/O devices can be changed by the operator during program operation. During an initial phase of program operation, a parameter selection module lets the operator choose a set of communications options to achieve compatibility with a host computer system. For convenience, a set of default conditions can be accepted if suitable for a particular communications link. Otherwise, the parameter module (via prompts on the CRT) permits the operator to select the following:

   - the desired baud rate
   - the type of parity
   - the number of data bits per character
   - the number of stop bits

   and to indicate how the host computer system normally reacts interactively in the following ways:

   - whether the host system does or does not echo each received character

   - whether the host system, upon receipt of a line of data denoted by a carriage-return character, automatically supplies only a line feed character, or supplies a line feed character followed by one or more characters, or supplies no characters.

2. If interested in additional information regarding Wang-developed software systems which can be used with a Synchronous/Asynchronous Communications Controller, contact the Wang Sales Office in the area where a Wang system is being used.

APPENDIX A

ASCII CODE SET

The Wang system character set is defined by 8-bit codes of the form $b_8b_7b_6b_5b_4b_3b_2b_1$, where $b_8=0$ and the bits b_7 through b_1 correspond to the ASCII (American Standard Code for Information Interchange) character set which has 128 assignment positions, as shown in Table A-1.

Wang CRT's and printers use the ASCII code set, but some units may not display all the graphic characters shown in Table A-1. In some cases, substitute graphic characters may be displayed by a Wang peripheral. For details, refer to the manual which accompanies a particular peripheral.

Table A-1. ASCII Code*

Low-order: 4-bits / High order: 4 bits / hex-digit	0_{0_0}	0_{0_1}	0_{1_0}	0_{1_1}	1_{0_0}	1_{0_1}	1_{1_0}	1_{1_1}
0	0	1	2	3	4	5	6	7
0000	0	NUL 0	SOH 1	STX 2	ETX 3	EOT 4	ENQ 5	ACK 6	BEL 7
0001	1	DLE 16	DC1 17	DC2 18	DC3 19	DC4 20	NAK 21	SYN 22	ETB 23
0010	2	Space 32	! 33	" 34	# 35	$ 36	% 37	& 38	(apos.) ' 39
0011	3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
0100	4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
0101	5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
0110	6	grave accent 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
0111	7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119

Low-order: 4-bits / High order: 4 bits / hex-digit	1_{0_0}	1_{0_1}	1_{1_0}	1_{1_1}	1_{0_0}	1_{0_1}	1_{1_0}	1_{1_1}
0	8	9	A	B	C	D	E	F
0000	0	BS 8	HT 9	LF 10	VT 11	FF 12	CR 13	SO 14	SI 15
0001	1	CAN 24	EM 25	SUB 26	ESC 27	FS 28	GS 29	RS 30	US 31
0010	2	(40	) 41	* 42	+ 43	(comma) , 44	(dash) - 45	(period) . 46	/ 47
0011	3	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
0100	4	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
0101	5	X 88	Y 89	Z 90	[91	\ 92	] 93	↑ 94	(underline) _ 95
0110	6	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
0111	7	x 120	y 121	z 122	{ 123	¦ 124	} 125	~ 126	DEL 127

***Numbers in the lower right corner of each box represent the decimal equivalent of the binary and the hexadecimal code for the character shown in the box, e.g., A = $(41)_{16}$ = $(01000001)_2$ = $(65)_{10}$.**

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 B

SPECIFICATIONS

Power Requirements

Supplied by the CPU.

Electrical Connection

A 25-pin RS-232-C, CCITT V.24 compatible female plug facilitates hookup of a modem.

Cable

A 12-foot (3.6m) cable, equipped with 25-pin RS-232-C compatible male connectors on each end, is supplied as an accessary.

Synchronous Communications Features and Compatible Modems

Determined by Wang-developed software acquired for use with the controller.

Asynchronous Transmission Rates

50, 75, 100, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200 or 9600 bits per second.

Asynchronous Character Format Options

Parity: odd, even, or no parity.
Number of Data Bits: 5,6,7 or 8.
Number of Stop Bits: 1, 1.5 or 2.

Asynchronous Communication Mode

Full or half duplex.

Compatible Modems for Asynchronous Applications

Bell 103 or 202 type, or equivalent.
Null modem, available from Wang, for direct communications link.

Standard Warranty Applies

APPENDIX C

USING A NULL MODEM

As indicated in Section 1.3, the modem used with Wang's Synchronous/Asynchronous Communications Controller for data communications over telephone lines must satisfy two requirements:

1. it must be compatible with the communications software being used in the Wang system and

2. it must match the characteristics of the modem at the remote site.

Modems are not available from Wang Laboratories for communications with remote sites.

On the other hand, for data communications over direct cables, Wang Laboratories does produce null modems. A null modem is a small, dual-plug device which ensures that the pin assignments in the cables and connectors directly linking two systems meet the standards recommended by the Electronic Industries Association RS-232-C, CCITT V.24 specifications.

The Model 2228N Null Modem is designed for use with either the Model 2228B or the Option 62B version of the Synchronous/Asynchronous Communications Controller. One null modem and a pair of RS-232-C, CCITT V.24 cables are needed when directly connecting the Wang system's communications controller to a host computer, a terminal, or another Wang system.

Since the 2228N null modem has polarity, care must be exercised when connecting two systems. The plugs at opposite ends of the null modem are not labeled individually, but the label reading 2228N is near one end and thereby distinguishes one plug from the other. The plug nearer the label 2228N should receive the cable leading from the conector on the Synchronous/Asynchronous Communications Controller in the Wang system. The other plug on the null modem should receive the cable leading from a host computer, a terminal, or a second Wang computer system.

NOTE:

When the 2228N null modem is used in conjunction with the Synchronous/Asynchronous Communications Controller operating with asynchronous software, the reverse channel capability is disabled.

EQUIPMENT MAINTENANCE

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 initiated by the customer.

A Maintenance Agreement protects your investment and offers the following benefits:

1. Preventive Maintenance -

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

2. Fixed Annual Cost -

   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 obtained from your local Sales-Service Office.

> NOTE:
>
> Wang Laboratories, Inc. does not honor guarantees or Maintenance Agreements for any equipment modified by a user. Damage to equipment incurred as a result of user modification is the financial responsibility of the user.

INDEX

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

700-4670A

TITLE OF MANUAL **SYNCHRONOUS/ASYNCHRONOUS COMMUNICATIONS CONTROLLER USER MANUAL (MODEL 2228C, MODEL 2228B, OR OPTION 62B)**

COMMENTS:

Fold

Fold

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

WANG

Fold

FIRST CLASS
PERMIT NO. 16
Lowell, Mass.

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

— POSTAGE WILL BE PAID BY —

WANG LABORATORIES, INC.
ONE INDUSTRIAL AVENUE
LOWELL, MASSACHUSETTS 01851

Attention: Technical Writing Department

Fold

Cut along dotted line.

Printed in U.S.A.
13-1019

United States

Alabama
Birmingham
Mobile

Alaska
Anchorage

Arizona
Phoenix
Tucson

California
Culver City
Fountain Valley
Fresno
Inglewood
Sacramento
San Diego
San Francisco
Santa Clara
Ventura

Colorado
Englewood

Connecticut
New Haven
Stamford
Wethersfield

District of Columbia
Washington

Florida
Miami
Hialeah
Jacksonville
Orlando
Tampa

Georgia
Atlanta
Savannah

Hawaii
Honolulu

Idaho
Idaho Falls

Illinois
Chicago
Morton
Park Ridge
Rock Island
Rosemont

Indiana
Indianapolis
South Bend

Kansas
Overland Park
Wichita

Kentucky
Louisville

Louisiana
Baton Rouge
Metairie

Maryland
Rockville
Towson

Massachusetts
Billerica
Boston
Burlington
Chelmsford
Lawrence
Littleton
Lowell
Tewksbury
Worcester

Michigan
Kentwood
Okemos
Southfield

Minnesota
Eden Prairie

Missouri
Creve Coeur

Nebraska
Omaha

Nevada
Las Vegas
Reno

New Hampshire
Manchester

New Jersey
Toms River
Mountainside
Clifton

New Mexico
Albuquerque

New York
Albany
Buffalo
Fairport
Lake Success
New York City
Syracuse

North Carolina
Charlotte
Greensboro
Raleigh

Ohio
Cincinnati
Cleveland
Middleburg Heights
Toledo
Worthington

Oklahoma
Oklahoma City
Tulsa

Oregon
Eugene
Portland

Pennsylvania
Allentown
Camp Hill
Erie
Philadelphia
Pittsburgh
Wayne

Rhode Island
Cranston

South Carolina
Charleston
Columbia

Tennessee
Chattanooga
Knoxville
Memphis
Nashville

Texas
Austin
Dallas
Houston
San Antonio

Utah
Salt Lake City

Vermont
Montpelier

Virginia
Newport News
Norfolk
Richmond

Washington
Richland
Seattle
Spokane
Tacoma

Wisconsin
Brookfield
Madison
Wauwatosa

International Offices

Australia
Wang Computer Pty., Ltd.
Adelaide, S.A.
Brisbane, Qld.
Canberra, A.C.T.
Darwin N.T.
Perth, W.A.
South Melbourne, Vic 3
Sydney, NSW

Austria
Wang Gesellschaft, m.b.H.
Vienna

Belgium
Wang Europe, S.A.
Brussels
Erpe-Mere

Canada
Wang Laboratories (Canada) Ltd.
Burnaby, B.C.
Calgary, Alberta
Don Mills, Ontario
Edmonton, Alberta
Hamilton, Ontario
Montreal, Quebec
Ottawa, Ontario
Winnipeg, Manitoba

China
Wang Industrial Co., Ltd.
Taipei
Wang Laboratories Ltd.
Taipei

France
Wang France S.A.R.L.
Paris
Bordeaux
Lyon
Marseilles
Nantes
Strasbourg
Toulouse

Great Britain
Wang (U.K.) Ltd.
Richmond
Birmingham
London
Manchester
Northwood Hills

Hong Kong
Wang Pacific Ltd.
Hong Kong

Japan
Wang Computer Ltd.
Tokyo

Netherlands
Wang Nederland B.V.
IJsselstein
Gronigen

New Zealand
Wang Computer Ltd.
Auckland
Wellington

Panama
Wang de Panama (CPEC) S.A.
Panama City

Singapore
Wang Computer (Pte) Ltd.
Singapore

Sweden
Wang Skandinaviska AB
Stockholm
Gothenburg
Malmo

Switzerland
Wang A.G.
Zurich
Basel
Geneva

Wang Trading A.G.
Zug

United States
Wang International Trade, Inc.
Lowell, Mass.

West Germany
Wang Laboratories, GmbH
Frankfurt
Berlin
Cologne
Dusseldorf
Essen
Freiburg
Hamburg
Hannover
Kassel
Munich
Nurnberg
Saarbrucken
Stuttgart

International Representatives

Abu-Dhabi
Argentina
Bahrain
Bolivia
Brazil
Canary Islands
Chile
Colombia
Costa Rica
Cyprus
Denmark
Dominican Republic
Ecuador
Egypt
El Salvador
Finland
Ghana
Greece
Guatemala
Haiti
Honduras
Iceland
India
Indonesia
Ireland
Israel
Italy
Jamaica
Japan
Jordan
Kenya
Korea
Kuwait
Lebanon
Liberia
Malaysia
Malta
Mexico
Morocco
Nicaragua
Nigeria
Norway
Paraguay
Peru
Phillippines
Portugal
Saudi Arabia
Scotland
Spain
Sri Lanka
Sudan
Syria
Thailand
Turkey
United Arab Emirates
Venezuela

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

Printed in U.S.A.
700-4670A
1-80-3M
Price: see current list