CS

Asynchronous Communications User Guide

For Model 2236MXE Terminal Processor and Option W Terminal Processor

2nd Edition — August 1987

700-8098A

WANG

WANG LABORATORIES, INC.
ONE INDUSTRIAL AVE., LOWELL, MA 01851 TEL. (617) 459-5000, TWX 710-343-6769, TELEX 172108

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, lease, or license agreement by which the product was acquired, nor increases in any way Wang's liability to the customer. In no event shall Wang or its subsidiaries be liable for incidental or consequential damages in connection with or arising from the use of the product, the accompanying manual, or any related materials.

Software Notice

All Wang Program Products (software) are licensed to customers in accordance with the terms and conditions of the Wang Standard Software License. No title or ownership of Wang software is transferred, and any use of the software beyond the terms of the aforesaid license, without the written authorization of Wang, is prohibited.

Warning

This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device, pursuant to Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

CONTENTS

FIGURES

TABLES

PREFACE

This manual describes the asynchronous communication capabilities of the Model 2236MXE and Option W Terminal Processors.

Chapter 1 describes the 2236MXE/Option W asynchronous communication capabilities and some important modem considerations for communication applications. Chapter 2 describes programming techniques related to the communication capabilities of the 2236MXE/Option W Terminal Processors.

Readers of this manual should be familiar with the *Multiuser BASIC-2 Language Reference Manual*.

CHAPTER 1
COMMUNICATION CAPABILITIES

OVERVIEW

The Wang Model 2236MXE or Option W Terminal Processor is a terminal interface device. The 2236MXE version can be used with CS, MicroVP, 2200MVP, and 2200LVP systems; the Option W version is used with the 2200SVP system. The 2236MXE version provides four terminal ports; the Option W version provides three terminal ports. The 2236MXE/Option W terminal processor is generally used for communications between Wang terminals and the central processor. It requires the @MXEO file contained in release 3.0 or greater of the BASIC-2 operating system.

The 2236MXE/Option W terminal processor is also capable of serving as a multiport, asynchronous, communications controller. When used as a communications controller, the 2236MXE/Option W is compatible with the Electronic Industries Association (EIA) RS-232-C standard and the Consultative Committee on International Telephony and Telegraphy (CCITT) V.24 standard. With the specialized telecommunications (TC) software supporting the terminal processor when it functions as a controller, you can alternatively choose any port (except system port 1) to act as an asynchronous communications controller. Via the port, you can attach compatible transmission equipment locally or remotely to the terminal processor.

The $GIO statement in BASIC-2 is used for program control of data transmission and reception via the 2236MXE/Option W port. Interaction between the MXE board and the BASIC-2 program utilizes a Communications Control Vector (CCV) to establish a configuration, and a Communications Status Vector (CSV) to show status. There are references to the CCV and the CSV in Chapter 1, and detailed descriptions of them in Chapter 2.

The connection of a 2236MXE/Option W port to a modem requires a 25-pin RS-232-C/V.24-compatible cable. (RS-232-C/V.24-compatible cables are available from Wang.) Communication applications require a modem 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 the section in this chapter entitled "Modem Considerations" for information that relates to the asynchronous communication capabilities of the 2236MXE/Option W terminal processors.

The ports on a 2236MXE/Option W terminal processor can also be used for direct connection of RS-232-C compatible, asynchronous, transmission equipment such as graphic display terminals, analytical equipment, and laboratory instrumentation in general. To attach such equipment, a null modem (available from Wang) may be required.

The 2236MXE/Option W terminal processor has an integrated microprocessor, and each port has multicharacter input/output buffers to simplify TC control procedures and reduce CPU processing requirements. The BASIC-2 operating system loads the TC-port control software into the 2236MXE/Option W microprocessor at system configuration time, in a step that is transparent to the user. The downloaded software supports the following:

- Data buffering
- Code translation
- Substitution for characters received in error
- Communications control, e.g., monitoring CPU ready/busy conditions, monitoring modem signals, and 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
- Automatic transmission delays following several specified characters

The standard and special features of the 2236MXE/Option W enable a Wang system to be programmed to transmit and receive data using the line discipline of a variety of asynchronous CRT and mechanical printer terminals.

Additionally, the 2236MXE/Option W's random access memory is used for storage of initialization information for each port (including code translation tables and a CCV), storage of current status information, and input/output data buffering. The desired transmission rate, communication mode, character format options, and special operations for a particular application are selected under program control by specified values in a CCV. The vector is loaded into the 2236MXE/Option W by a $GIO statement in the user's application program operating in the CPU.

A selectable-speed clock on each port of the 2236MXE/Option W supports serial asynchronous transmission and reception at line speeds from 50 to 9600 bits per second (bps).

The 2236MXE/Option W supports the following transmission modes:

- Full-duplex mode (independent transmission two ways simultaneously)
- Full-duplex mode with automatic deletion of null characters received
- Half-duplex mode (independent transmission one way at a time alternately)
- Half-duplex mode with automatic deletion of null characters received

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

Other byte positions in the CCV are used to select the character format. The following options are available:

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

The communication capabilities of the 2236MXE/Option W are the same as for the Wang Model 2227B/Option 62 Buffered Asynchronous Communications Controller, except that reverse channel in half-duplex transmission mode and data transmission by means of the BASIC-2 statement PRINT, PRINTUSING, or MAT PRINT are not supported. The following line speeds also are not supported: 1800, 3600, and 7200 bps.

The following Wang TC-software packages are supported by the 2236MXE/Option W terminal processor: Asynchronous 1 (ASC 1), Release 7.0 or greater, and the Remote Control and Maintenance (RCM) system, Release 1.3 or greater. Asynchronous 1 provides the capability to emulate the TTY and 2741 asynchronous protocols. The RCM system allows an RCM-equipped 2200 Series computer to perform extensive maintenance and program functions on one or more remote systems. Support of ASC 1 or the RCM system requires Release 2.5 or greater of the BASIC-2 operating system.

The communication features of the 2236MXE/Option W terminal processor are numerous; descriptions of particular features are presented in this chapter. Programming techniques are presented in Chapter 2; a sample program is included.

INSTALLATION

Installation of the 2236MXE/Option W terminal processor is the responsibility of a Wang service representative. Do not attempt to install the 2236MXE/Option W terminal processor because any such attempt may damage the equipment and void the warranty.

If the 2236MXE/Option W is to be used for point-to-point, dial-up asynchronous telecommunications applications, an RS-232-C/V.24-compatible cable should be plugged into a suitable modem and into the selected port on the 2236MXE/Option W terminal processor. (RS-232-C/V.24-compatible cables are available from Wang.) A Wang service representative is responsible only for installation of a Wang modem.

Alternatively, if the 2236MXE/Option W is to be used to interface RS-232-C/V.24-compatible asynchronous transmission equipment such as a laboratory instrument, the other end of the cable may need to be plugged into a null modem (available from Wang) or may be suitable for direct connection to the non-Wang equipment. Installation or interfacing of non-Wang equipment is not the responsibility of Wang. The section that discusses Connector Pin Assignments provides information regarding the interface connector pin assignments and voltage levels.

MODEM CONSIDERATIONS

The modem used with the 2236MXE/Option W may be rented from the telephone company serving the locality where a Wang system is installed or may be purchased from any of several modem vendors, including Wang.

Federal Communications Commission (FCC) regulations (Part 68) specify that before connecting a device such as a modem to the switched telephone network, you must provide the local telephone company with the name of the manufacturer of the device to be attached, the model number, FCC registration number, and ringer equivalence number of the device to be connected.

Before installing the modem, you must plan the location of the Wang system to ensure its proximity to the modem. The RS-232-C standard recommends use of short cables (less than 50 feet or 15 meters) between data terminal equipment and communications equipment. Longer cable distances are possible for operations at a lower range of transmission rates in an environment relatively free of electromagnetic interference.

The microprocessor on the 2236MXE/Option W can sense the value of the following modem signals:

- Received Data on Pin 3
- Clear to Send on Pin 5
- Data Set Ready on Pin 6
- Received Line Signal Detector on Pin 8

The microprocessor can set the level of the following modem signals:

- Data Terminal Ready on Pin 20
- Request to Send on Pin 4
- Transmitted Data on Pin 2

A modem with appropriate full-duplex, asynchronous capabilities, such as the Wang WA3451 Asynchronous/Synchronous Modem or a Wang Telemodem, can be used with the 2236MXE/Option W. Other RS-232-C-compatible modems commonly used with asynchronous terminals having transmission rates in the range covered by the 2236MXE/Option W may also prove suitable.

For asynchronous operations, the WA3451 modem has three functional modes: 103J-compatible mode, 212A-compatible mode, and 3400 mode. The 103J-compatible mode supports asynchronous communications with a Bell 103J modem at line speeds up to 300 bps. The 212A-compatible mode supports asynchronous communications with a Bell 212A at a line speed of 1200 bps. The 3400 mode supports asynchronous communications with another WA3451 modem, a Racal-Vadic 3400 series modem, or an Anderson-Jacobson 1200 modem at either a line speed of 1200 bps, or any standard commercial line speed of 300 bps and below.

The WA3451 modem can be used for three types of communication operations over a switched telephone line: manually originated calls, manually answered calls, and automatically answered calls.

Note: *Modems used at both ends of a communications link must be of similar type. For example, if a Wang WA3451 modem operating in the 103J-compatible mode is used at one end, a Wang WA3451, Bell 103J, or equivalent modem must be used at the other end. When a non-Wang modem is considered, a switched Clear to Send signal is mandatory (Pin 5); otherwise, the power-up diagnostics erroneously indicate a problem with a Wang terminal.*

If acoustic couplers are used at both ends of a communications link, one must have the Originate feature and the other must have the Answer feature. Ideally each should have both features.

CONNECTOR PIN ASSIGNMENTS

This section provides information for readers responsible for interfacing non-Wang equipment to a Wang system via the 2236MXE/Option W terminal processor.

The 2236MXE/Option W conforms to the nationally recognized EIA RS-232-C and the internationally recognized CCITT V.24 standards for voltage levels and pin connections. The signal polarity and the voltage of driven and detected signals are as follows:

Logic Level	Applied Voltage	Detected Voltage
0 or ON (Spacing)	+8 vdc	+5 to +15 vdc
1 or OFF (Marking)	-8 vdc	-5 to -15 vdc

The interface connector pin assignments are listed in Table 1-1, with both the EIA and the CCITT designations given for the circuit associated with each pin. The signal descriptions and sources are also included in the table.

Table 1-1. Connector Pin Assignments

Pin	EIA	CCITT	Signal Description	Source
1	AA	101	Protective Ground	
2	BA	103	Transmitted Data	Controller
3	BB	104	Received Data	Modem
4	CA	105	Request to Send	Controller
5	CB	106	Clear to Send	Modem
6	CC	107	Data Set Ready	Modem
7	AB	102	Signal Ground	
8	CF	109	Received Line Signal	Modem

(continued)

Table 1-1. Connector Pin Assignments (continued)

Pin	EIA	CCITT	Signal Description	Source
			Detector[a]	
9			Not Used	
10			Not Used	
11			Not Used	
12			Not Used	
13			Not Used	
14			Not Used	
15			Not Used	
16			Not Used	
17			Not Used	
18			Remote Analog Loopback[a]	Controller
19			Not Used	
20	CD	108.2	Data Terminal Ready[a]	Controller
21			Remote Digital Loopback[a]	Controller
22	CE	125	Ring Indicator[a]	Modem
23	CH/CI	111/112	Data Signalling Rate Selector[a]	Controller/Modem
24			Not Used	
25			Not Used	

[a] These pins are not connected on port 1 of an SVP Option W because that port is intended for a local terminal only. Ports 2 and 3 use these signals.

ASYNCHRONOUS TRANSMISSION AND RECEPTION

In asynchronous transmission, each character is framed by start and stop elements as shown in Figure 1-1. 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.

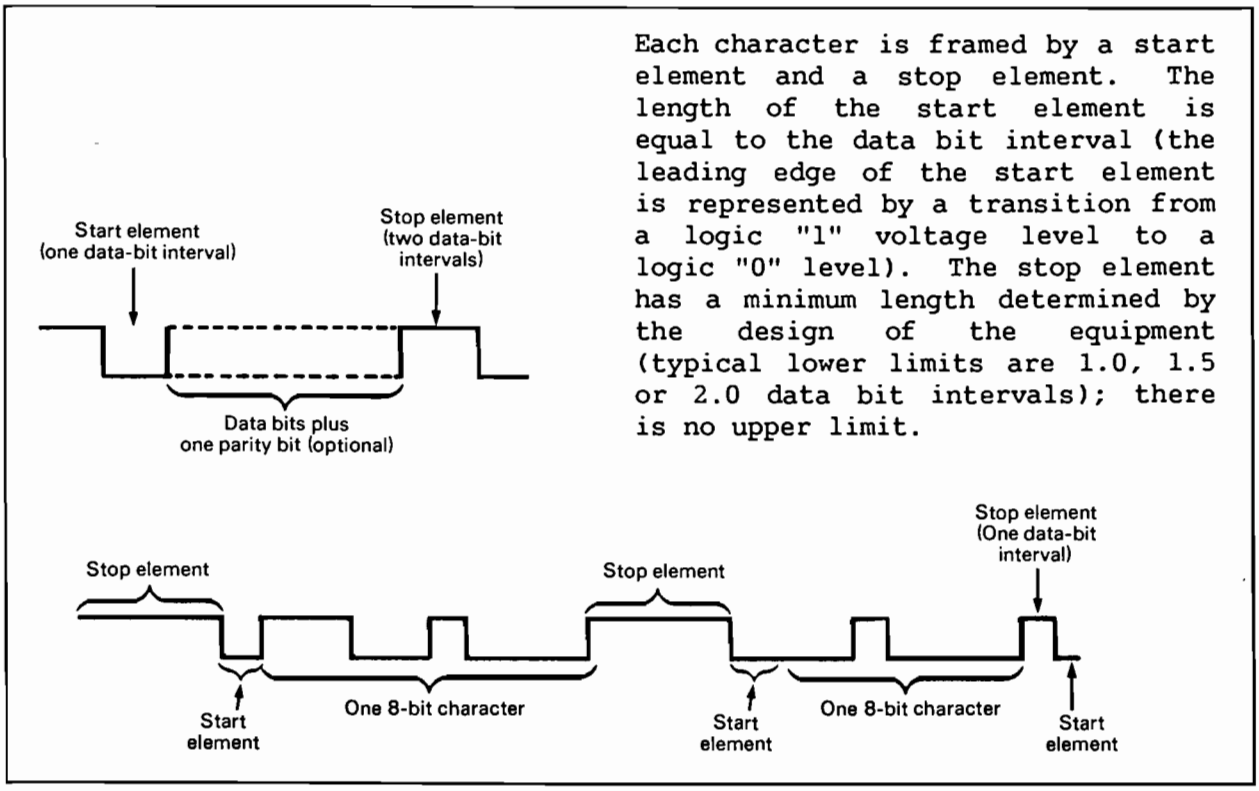

Figure 1-1. Asynchronous Data Transmission

Depending on the design of the transmitting equipment, the data-bit interval is a fixed value or one of several possible values if the transmission rate for the equipment is selectable. 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 1-1.) 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 data bits transmitted is a fixed value or one of several possible values, depending on the equipment used.

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 bit and start bit intervals. The voltage level may be a logic "0" or "1" depending upon the type of parity (odd or even) and also upon the number of 1s 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, a fixed value, or one of several possible values, depending upon the design of the transmitting equipment.

Character Transmission

Wang's 2236MXE/Option W terminal processor transmits each character over a port selected for TC 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. For example, if the number specified is 6, bits 1 through 6 of the character are transmitted, and the remaining two bits are discarded.

3. If no parity is specified, Step 3 is omitted. If odd or even parity is specified in the CCV, 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; for odd parity, the total number of 1s 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; for even parity, the total number of 1s in the data bits, plus the parity bit, is an even number.

4. The Transmitted Data signal is set to "1" for a minimum interval equal to 1, 1.5, or 2 bit-times depending on the number of stop bits specified in the CCV.

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

Character Reception

A 2236MXE/Option W port 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. Beginning with the low-order bit, the bits in the character being received are set successively to correspond to the sampled values. The number of data bit samples taken by the 2236MXE/Option W equals the number of data bits specified in the CCV. If the number of samples is fewer 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 CSV 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 for 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 and a designated bit in the status vector is set to 1.

Note: *For each application, the transmission rate, number of data bits, type of parity, and number of stop bits specified for a 2236MXE/Option W port must match the specifications for equipment in use at the other end of a communications link.*

DATA BUFFERING

Each port of the 2236MXE/Option W terminal processor has two multicharacter data buffers, a 256-byte transmit buffer and a 256-byte receive buffer. The 2236MXE/Option W uses these buffers to temporarily store data during its transmission/reception operations. A modem attached to one of these ports can overlap data while the CPU is performing input/output operations to the I/O peripherals for a communications application.

For example, after the CPU sends a data string to the 2236MXE/Option W, 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 2236MXE/Option W is performing such tasks as code translation, character formatting, and transmission to the modem.

Note: *If the transmit buffer of a port becomes full while the CPU is sending data to the buffer, the data transfer rate from the CPU to the 2236MXE/Option W automatically slows to the rate at which characters are being transmitted from the buffer. No characters are lost.*

The 2236MXE/Option W is free to receive a data string, perform such operations as code translation, and store the data in the receive buffer of a port, while the CPU is performing an independent task such as sending data to a designated peripheral.

Note: *If characters are received when the receive buffer of a port is full, a buffer overflow condition occurs and the appropriate error bit is set in the CSV. No other action is taken by the 2236MXE/Option W, and the overflow bytes are lost.*

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 CCV) is automatically supplied by the 2236MXE/Option W, and the appropriate error bit is set in the port's CSV. 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.

TRANSMISSION DELAYS FOLLOWING SPECIFIED CHARACTERS

When sending data to a mechanical printer terminal such as an IBM 2741, time delays are needed after transmission of TAB and RETURN characters to allow the printing element sufficient time to reach the proper position without loss of subsequent characters.

A special feature of the 2236MXE/Option W removes the necessity for the user's application program to introduce time delays following transmission of special characters. Bytes 11 through 18 of the communications control vector can be used, in four 2-byte groups, to define up to four special characters and the transmission delay associated with each special character. During data transmission, the 2236MXE/Option W automatically delays for the specified time following transmission of any character matching one of the specified characters.

CODE TRANSLATION

The code translation feature of the 2236MXE/Option W allows data interchange between the CPU and the 2236MXE/Option W in the ASCII code (American Standard Code for Information Interchange) native to the CPU, regardless of the transmission/reception code for a particular application.

Each port has space reserved in the 2236MXE/Option W for two 256-byte code translation tables, a *transmit code translation table* and a *receive code translation table*. Specification of such tables is optional. Translation tables, supplied in the user's application program operating in the CPU, must be loaded into the 2236MXE/Option W by appropriate $GIO statements in the program. (See Chapter 2.)

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 disabled.

During transmission, a character sent from the CPU to the 2236MXE/Option W becomes an 8-bit index for a table lookup in the *transmit code translation table*. An 8-bit character obtained from the table is placed in the transmit buffer. However, if byte 3 of the CCV specifies less than 8 data bits per character, only the relevant low-order bits of each character are actually transmitted.

During reception, a character received by the 2236MXE/Option W 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, i.e., HEX(00), and the high-order hexdigit in byte position 2 in the CCV has the value 3, 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 the full-duplex and half-duplex operations supported by the 2236MXE/Option W terminal processor. (See Table 2-1.)*

INSERTION AND REMOVAL OF SHIFT CHARACTERS

For applications involving data transmission and reception in a code set that utilizes shift characters, e.g., a Baudot code set or an IBM 2741 code set, a special feature of the 2236MXE/Option W removes the necessity for the user's program operating in the CPU to handle insertion and removal of shift characters. Instead, the upshift and the downshift characters are defined in bytes 7 and 8 of the CCV. 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 CCV. Then, to activate automatic insertion and removal of shift characters, code translation tables are suitably defined and loaded into the 2236MXE/Option W.

During data transmission, the 2236MXE/Option W examines and interprets the two high-order bits of each translated character in the transmit buffer as Shift Status bits as shown in Table 1-2.

Table 1-2. Shift Status Bits

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 having different Shift Status bits. 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 being transmitted per character is only the low-order 5 or 6 bits if byte 3 in the CCV has been appropriately specified.

During reception, the 2236MXE/Option W sets the value of the shift status bit (high-order eighth bit) of each received character (before code translation) according to the most recently received shift character. The 1 bit represents an upshifted character and the 0 bit represents a downshifted character.

Note: *Before code translation, each incoming character is compared to the shift characters in bytes 7 and 8 of the communications control vector. When a shift character is found, the shift status bit in the subsequent received characters is set accordingly, and the shift character is discarded. The subsequent received characters are then code translated and placed in the receive buffer.*

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., RETURN codes. This feature is particularly convenient for applications where it is not necessary to display the data while it is being received.

Any number of characters can be defined as end-of-record characters, thereby permitting the 2236MXE/Option W to divide a received data stream into records while eliminating the need for the user's program operating in the 2200 CPU to perform the task.

To activate the end-of-record detection feature, byte 6 in the CCV must be set to HEX(01). Also, a suitably defined *receive code translation* table must be loaded into the 2236MXE/Option W. 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 2236MXE/Option W maintains a count of the number of end-of-record characters currently stored in the active port's receive buffer. The count is maintained in byte 5 of the CSV (see Table 2-2).

With an appropriate $GIO statement (see Chapter 2), the user's application program can read the status vector into the CPU and subsequently test the status information to ensure the availability of a complete record before requesting transfer of buffered data via a $GIO statement which performs data transfer from the 2236MXE/Option W to the CPU.

When data is actually transferred from the port'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 utilized 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 CCV should be set to HEX(00) to disable end-of-record detection.*

MONITORING RECEIVED TIMEOUTS

The 2236MXE/Option W has the capability to monitor received timeouts. Byte 5 in the CCV is used to set the binary value of the timeout in units of 0.1 second. For example, the minimum timeout condition, 0.1 second, is specified by storing HEX(01) in byte position 5. The maximum timeout condition, 25.5 seconds, is specified by storing HEX(FF) in byte position 5. On the other hand, storing HEX(00) in byte position 5 disables the monitoring feature for received timeouts.

If a timeout interval is specified, the 2236MXE/Option W maintains a received data timeout countdown in byte 6 of the CSV (see Chapter 2). Every 0.1 second during reception, byte 6 is decreased by 1 until it reaches 0.

SENDING AND DETECTING BREAK SIGNALS

The 2236MXE/Option W has the capability to send and detect break signals under control of the user's program operating in the CPU. Bytes 9 and 10 in the CCV are used to define the break signal transmission and detection intervals, respectively, in units of 10 milliseconds. For example, HEX(14) stored in byte position 9 defines an interval equal to 200 milliseconds for transmitted break signals. Similarly, HEX(11) stored in byte position 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 of the CCV, it is necessary to use the low-order hexdigit position in byte 2 to enable or disable the break signal. The hexdigit 0 disables the break signal and the hexdigit 1 enables the break signal so that data can be transmitted and received.

Transmission of a break signal by the 2236MXE/Option W involves inverting the level of the Transmitted Data modem signal for an interval defined by byte 9 of the CCV.

Detection of a break signal occurs when the 2236MXE/Option W senses that the level of the *Received Data* modem signal is being continuously inverted for an interval at least as long as the interval defined by byte 10 of the CCV.

Note: *Detection of a break signal causes the Break Signal Received bit in the CSV to be set. No other action is taken by the 2236MXE/Option W.*

CHAPTER 2
PROGRAMMING TECHNIQUES

GENERAL CONSIDERATIONS

When writing a communications application program for the 2236MXE/Option W terminal processor, you should organize the program in distinct modules designed to achieve initialization and communication functions. The communications module could overlay the initialization module or, if preferred, the two modules could coexist in the program.

An initialization module defines the Communications Control Vector (CCV) and assigns particular values to byte positions denoting information such as the desired transmission rate in bits per second, the number of data bits per character, the type of parity (if any), and the number of stop bits per character. Other values in the control vector indicate whether the 2236MXE/Option W is to activate such features as break detection, monitoring of received timeouts, full- or half-duplex operation with or without automatic removal of null characters, and substitution of a special character for characters received with parity or framing errors. The module also supplies the $GIO statement needed to load the control vector into the 2236MXE/Option W.

An initialization module also defines transmit and receive code translation tables, if required for the application, and supplies the $GIO statements needed to load such tables into the 2236MXE/Option W. Additionally, the initialization module might define and initialize variables required by the communications module even though the variables are stored in the CPU rather than the 2236MXE/Option W memory.

Note: The translation tables, if any, and the CCV must be loaded into the 2236 MXE/Option W via a $GIO statement having the appropriate microcommand sequence.

SPECIFYING THE COMMUNICATIONS CONTROL VECTOR (CCV)

Space is reserved in the random access memory of the 2236MXE/Option W for each port to have a CCV. The user configures a particular port for communications, as described in the section entitled "CPU and 2236MXE/Option W Interaction Via $GIO Statements", and needs to define the CCV only once for a given application program. The format of the CCV is shown in Figure 2-1, and valid specifications for it are given in Tables 2-1 and 2-2. The CCV must be established with a length of 20 or 29 bytes depending on the specifications desired.

In the application program residing in the CPU, the CCV should be defined by a one-dimensional array having either 20 or 29 elements with one byte per element. For example, to represent the CCV by the array C$(), use

```
DIM C$(20)1 or C$(29)1
```

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

```
C$()=ALL(00)          or          INIT(00)C$()
```

before assigning values to particular elements in the array.

Then, as illustrated by the following statements, individual byte positions in the CCV can be assigned values other than binary zero to select the desired combination of options and define any special characters for the application.

Statement	Comment
C$(1) = HEX(17)	One stop bit; 300 bits per second
C$(2) = HEX(31)	Full 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

Exercise care when defining some special characters to choose a compatible value in the first three byte positions of the CCV. 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 Chapter 1 and Table 2-2.)

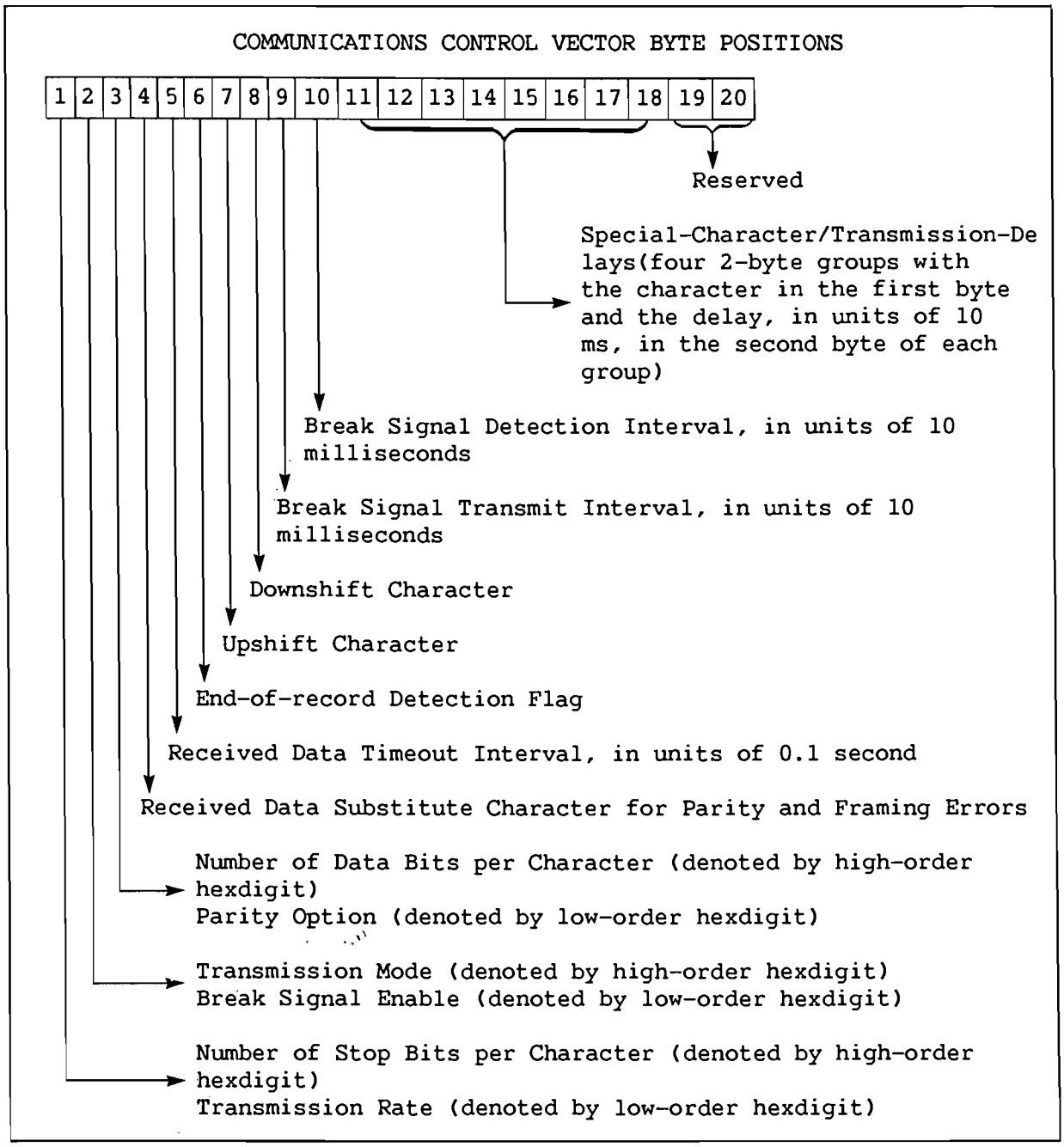

Figure 2-1. Communications Control Vector Format
CCV Bytes 1-20

Note: *See Tables 2-1 and 2-2 for valid value specification.*

Communications Control Vector Extension

Optional bytes 21 through 29 may be appended to the end of the communications control vector with no changes to already existing byte definitions. If you want XON/XOFF or flow control via DTR, you should load all bytes through byte 29. If you do not want to use these functions, you need only load the first 20 bytes of the control vector. Therefore, no old applications are affected by this extension.

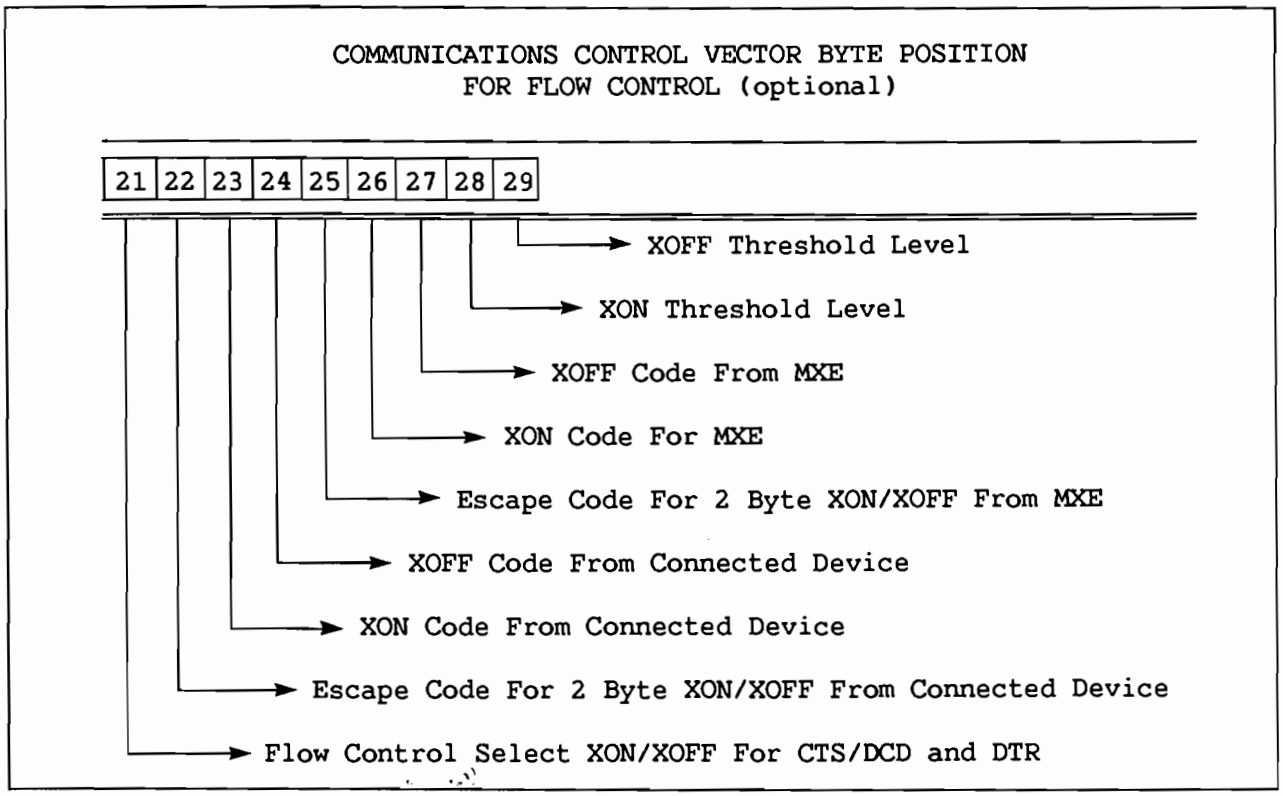

Figure 2-2. Communications Control Vector Extension Format CCV Bytes 21-29

Note: See Table 2-3 for valid value clarification.

Table 2-1. Valid Communications Control Vector Specifications
CCV Bytes 1-3

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 bits per second (bps) 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 = Undefined B = 2400 bps C = Undefined D = 4800 bps E = Undefined 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)
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

Note: *See Table 2-2 for CCV bytes 4-20 and Table 2-3 for CCV bytes 21-29.*

Table 2-2. Valid Communications Control Vector Specifications CCV Bytes 4-20

Byte	Byte Value[a]	Remarks
4	xy = Substitute character for parity/framing error	Each received character having a parity or framing error is replaced by the parity/framing errors designated character (Replacement occurs prior to code translation if translation tables are being used.) (See Chapter 1.)
5	xy = Timeout interval in units of 0.1 second	The specification in hexadecimal notation represents the timeout interval in units of 0.1 second, e.g., $(24)_{16} = (36)_{10}$ specifies an interval of 3.6 seconds. (See Chapter 1.)
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 in each code that represents an incoming end-of-record character. (See Chapter 1.)
7	xy = Upshift character	To enable shift code insertion and 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 10 or 11 as described in Chapter 1. The receive code translation table must allow for the 2236MXE/Option W's automatic setting (before translation) of the high-order bit to 1 for all incoming upshifted characters.
8	xy = Downshift character	

(continued)

Table 2-2. Valid Communications Control Vector Specifications CCV Bytes 4-20 (continued)

Byte	Byte Value[a]	Remarks
9	xy = Break signal transmit interval in units of 10 ms	To enable break signal transmission and detection, the low-order hexdigit in byte 2 of the CCV must be 1 (i.e., the break signal must be enabled.) If the bytes 9 and 10 are both in HEX(00), 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 Chapter 1.)
10	xy = Break signal detection interval in units of 10 ms	
11	xy = Special character	Up to four special characters can be defined in bytes 11, 13, 15, and 17. The delay associated with a particular character is specified in the following byte, in hexadecimal notation representing the interval in units of 10 milliseconds. The maximum delay, HEX(FF), is 2.5 seconds. During transmission, the 2236MXE/Option W automatically delays for the specified time following transmission of one of the specified characters. If a transmit code translation table is used, each special character must be defined with respect to its code after translation. If the automatic transmission delay capability is not desired, bytes 11 through 18 should each be set to HEX(00).
12	xy = Transmission delay in units of 10 ms	
13	Same as byte 11	
14	Same as byte 12	
15	Same as byte 11	
16	Same as byte 12	
17	Same as byte 11	
18	Same as byte 12	
19	Reserved	
20	Reserved	

[a]x and y each denote any hexdigit (0 through 9, A through F).

Bit positions in each byte are numbered from 8 (high order) to 1 (low order) and are referred to in terms of a bit mask.

Upper Hex Digit		Lower Hex Digit	
Bit Position	= Bit Mask	Bit Postion	= Bit Mask
8	80	4	08
7	40	3	04
6	20	2	02
5	10	1	01

A value 'xy' refers to all bit positions in a byte. A single hex digit 0-9 or A-F refers to all bit positions in either the upper or lower hex digit of a byte. For example, the xy value '2A' sets bits in bit positions 6, 4, and 2; or sets bits in mask positions 20, 08, and 02. For additional information on using the CCV extension refer to the text following Table 2-3.

Table 2-3. Valid Optional Communications Control Vector Specifications CCV Bytes 21-29

Byte	Bit Mask	High- and Low-Order Hexdigits	Bit	Meaning	
21	01	0y	1	1 = Select CTS/DCD as flow control from connected device.	H/W
	02		2	1 = Select DTR as flow control from the MXE controller.	H/W
	04		3	1 = Select XON/XOFF as flow control from connected device.	S/W
	08		4	1 = Select XON/XOFF as flow control from the MXE controller.	S/W

(continued)

Table 2-3. Valid Optional Communications Control Vector Specifications CCV Bytes 21-29 (continued)

Byte	Byte Value	Meaning
22	xy = Escape code for 2-byte escape sequences for XON/XOFF flow control from connected device.	If this byte is zero, a 1-byte flow control code is assumed. If this byte is not zero, the code specified must precede byte 23 or 24 for the 2-byte sequence to be recognized as a valid XON/XOFF sequence.
23	xy = XON code from connected device.	
24	xy = XOFF code from connected device.	
25	xy = Escape code for 2-byte escape sequences for XON/XOFF flow control from the MXE controller.	If this byte is zero, a 1-byte flow control code is assumed. If this byte is not zero, the code specified must precede byte 26 or 27 for the 2-byte sequence to be recognized as a valid XON/XOFF sequence.
26	xy = XON code from the MXE controller.	
27	xy = XOFF code from the MXE controller.	
28	xy = XON threshold level	XON threshold level. The MXEcontroller sends the XON byte(s) when the buffer level decreases to this count and there is a pending XOFF. When using DTR flow control, the MXE controller enables DTR at this level.

(continued)

Table 2-3. Valid Optional Communications Control Vector Specifications CCV Bytes 21-29 (continued)

Byte	Byte Value	Meaning
29	xy = XOFF threshold level	XOFF threshold level. The MXE controller sends the XOFF byte(s) when the buffer level increases to this count and there is a pending XON. When using DTR flow control, the MXE controller disables DTR at this level.

You should keep in mind the following points when using the optional CCV bytes.

1. If the XON byte has the same definition as the XOFF byte the MXE controller uses the defaults of hex 11 and hex 13, respectively.

2. Zero is a valid, though not recommended, flow control code.

3. If you select a 2-byte flow control sequence, valid XON/XOFF codes are deleted from the data stream. 2-byte sequences that are not valid flow controls are passed to the program.

4. When you are specifying the buffer empty levels, the values must be in the range of 1 to 254 (hex 1-FE), and the XOFF level must be greater than or equal to the XON level. If either value is out of range, or if the XON level is greater that the XOFF level, the MXE controller uses defaults of 32 and 200, respectively.

5. If you select XON/XOFF while half duplex is selected, the MXE controller forces full duplex mode during initialization.

6. When using XON/XOFF flow control, the MXE controller sends XON/XOFF controls even when the connected device has sent an XOFF code.

7. When using DTR flow control, the MXE controller accepts data from the connected device even when DTR is inactive. However, overrun may occur if data is received when DTR is low.

Note: *If DCD is low and you have selected CTS/DCD flow control, the MXE controller does not recognize data on the line, including XON/XOFF bytes, as valid.*

8. The MXE controller does not send XON/XOFF if CTS is low and CTS/DCD are selected as flow controls.

9. The MXE controller sends an XOFF byte on each of the following events:

   - the buffer level reaches the specified XOFF threshold
   - the buffer level reaches 254
   - a buffer overflow occurs

10. If reception is halted immediately after the MXE controller sends an XOFF character to the connected device, it is possible for a lockout condition to occur. To prevent this, the MXE controller sends an XON character when reception is enabled, CBS (08), if the buffer level is less than or equal to the XON level. If DTR is selected, the signal is lowered or raised depending on the XON level.

THE COMMUNICATIONS STATUS VECTOR (CSV)

Space is reserved in the random access memory of the 2236MXE/Option W for each port to have a 7- or 8- byte CSV whose byte and bit positions are used automatically, as shown in Table 2-2. The first three bytes of the CSV are cleared automatically whenever it is read and when the port's CCV is loaded into the 2236MXE/Option W from the CPU.

Flags are set in particular bit positions in the first three bytes of the CSV during 2236MXE/Option W operation. In bytes 4 and 5, the current number of characters in the receive buffer and the number of end-of-record characters are maintained as binary counts. Similarly, in byte 7, the current number of characters in the transmit buffer is maintained. Byte 6, on the other hand, is similar to a real-time clock whose value is initialized to the timeout interval specified in byte 5 of the CCV each time one of the following events occurs:

1. A $GIO *Start Receiving Data* operation begins.

2. A $GIO *Send Data* operation begins.

3. A $GIO *Send Then Receive Data* operation begins.

4. A line turnaround occurs during a $GIO *Send Then Receive Data* operation.

5. A character is received.

However, if the value in byte 6 of the CSV is not re-initialized by one of these operations, the value is decreased by 1 every 0.1 second until it reaches 0.

Whenever desired, the information in the CSV can be read (transferred to the CPU) by a $GIO statement having the appropriate microcommand sequence. After transfer to the CPU, CSV information can be tested, as required, by the application program.

CSV Extension

An optional eighth byte of the CSV contains the current state of all control signals supported by the MXE controller.

The two bits (40 and 80) indicate the current receive/transmit state in the respective direction. When XON/XOFF is selected, the actual flow control bytes being received or transmitted determine the setting of the two bits. If CTS/DCD and DTR are selected without XON/XOFF, bit 40 is set by the current state of CTS, and bit 80 is set by the current state of DTR.

To allow compatibility with older applications, you need input only the first seven bytes of the CSV.

Table 2-4. CSV Information

Byte	Bit Mask	Bit[a]	Meaning
1	01	1	1 = Break signal received
2	01	1	1 = Received Line Signal Detector modem signal on
	02	2	0 = Always zero
	04	3	1 = Data Set Ready modem signal on
3	01	1	1 = Receive parity error detected
	02	2	1 = Receive buffer overflow error detected
	04	3	1 = Receive framing error detected

(continued)

Table 2-4. CSV Information (continued)

Byte	Byte Value	Meaning
4	xy	Binary count of the number of characters in the receive buffer
5	xy	Binary count of the number of end-of-record characters in the receive buffer
6	xy	Received data timeout countdown
7	xy	Binary count of the number of characters in the transmit buffer

Byte	Bit Mask	Bit[a]	Meaning (1 = True)
8	01	1	DCD (Received Line Signal Detector) Pin 8
	02	2	RI (Ring Indicator) Pin 22
	04	3	DSR (Data Set Ready) Pin 6
	08	4	CTS (Clear To Send) Pin 5
	10	5	DTR (Data Terminal Ready) Pin 20
	20	6	RTS (Request To Send) Pin 4
	40	7	XOFF byte received by the MXE controller from connected device.
	80	8	XOFF byte sent by the MXE controller to connected device.

[a] Bit positions in each byte are numbered from 1 (low order) to 8 (high order).

CPU AND 2236MXE/OPTION W INTERACTION VIA $GIO STATEMENTS

To operate the 2236MXE/Option W terminal processor in communications mode, the user's application program residing in the CPU should configure a 2236MXE/Option W port for telecommunications. This step is needed because the port can also be configured for the attachment of a terminal, as described below. The program should also include $GIO statements with a port address and suitable microcommand sequences.

Port Configuration

A port address is defined as follows:

port address = /Add

where device-type A indicates the 2236MXE/Option W terminal processor and dd is a two-digit number 02, 03, ..., up to 16, indicating the system port number (terminal number). A CPU can accommodate up to four 2236MXE terminal processors, each with four ports. For example, /A05 is the first RS-232-C port on the second 2236MXE, and /A10 is the second RS-232-C port on the third 2236MXE.[1] However, a 2200SVP central processor can accommodate only one Option W terminal processor with three ports. /A02 is the second and /A03 is the third RS-232-C port on the Option W terminal processor.

A 2236MXE/Option W port must be configured as a TC port before it can be used for telecommunications. This is accomplished with the SELECT TC statement, defined as follows:

SELECT TC port-address

where port-address is the address of the port to be configured as a TC port. To select a port as a TC port, all of the following conditions must be true:

- The port must be on a 2236MXE/Option W terminal processor.
- The port must not be attached or assigned to any partition.
- The port must not be terminal port 1 (i.e., port address /A01).

To reconfigure the port as a terminal port, the SELECT TERMINAL statement is used:

SELECT TERMINAL port-address

If a TC port is opened and then reconfigured as a terminal port in the same partition, the port is automatically closed before it becomes a terminal port. (Opening and closing a TC port is discussed later.) However, if the TC port has been opened by another partition, the SELECT TERMINAL statement produces an error.

Selecting a port to its current configuration (i.e., a TC port to be a TC port or a terminal port to be a terminal port) does not produce an error.

[1] Refer to Appendix B for a list of port address designations.

The $OPEN and $CLOSE statements of the BASIC-2 language can be used with any CPU input/output port except a terminal port. These statements can be used to reserve exclusive access to (i.e., lock around) a TC port, after the port has been configured. (It is recommended that a TC port be opened as soon as it is configured to prevent another user from accidentally selecting it.) The following are examples of the use of $OPEN and $CLOSE statements:

```
$OPEN /A07
$OPEN 200, /A06
$CLOSE /A12
```

TC $GIO Statements

The format of the $GIO statements used to address a TC port is as follows:

$GIO port-address (GIO command)

where port-address is a direct or indirect port address and (GIO command) is the body of the $GIO statement, including the microcommand sequence. Most formats that are available within the body of the $GIO statement are legal, with the exception of address switching. Attempts to switch addresses within a $GIO statement cause a run-time error.

A list of valid microcommand sequences for 2236MXE/Option W operations is presented in Table 2-5.

Table 2-5. Microcommand Sequences for 2236MXE/Option W and CPU Interaction

2236MXE/Option W and CPU Interaction	Microcommand Sequence[a]	Remarks
Set CCV	4402 A000 440C	
Read CSV	4403 1020 02FF 03FF 1223 C620	
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 mode

(continued)

Table 2-5. Microcommand Sequences for 2236MXE/Option W and CPU Interaction (continued)

2236MXE/Option W and CPU Interaction	Microcommand Sequence[a]	Remarks
Transfer received data to CPU	4409 1020 02FF 03FF 1223 C620	For half- or full-duplex mode
Send data	440A A000 440C	For half- or full-duplex mode
Send, then receive data	440B A000 440C	For half-duplex mode
Stop transmitting	440C	For full-duplex mode
Continue transmitting	440D	For full-duplex mode
Reset controller	4580	For half- or full-duplex mode
Set signals	440F A000 440C or 440F A000 4400	

[a] A microcommand sequence can be specified directly or indirectly in a $GIO statement. If specified indirectly by assigning the sequence to a variable, the dimension of the variable must be large enough to ensure the presence of two trailing space characters that 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. (Refer to The Wang *BASIC-2 Language Reference Manual* for more information.)

Brief descriptions of each of the operations listed in Table 2-5 follow. A sample $GIO statement is shown for each operation; however, the 4-hexdigit code values used in the statement may differ and the variables may be given different names in a user's program.

Set Communications Control Vector (CCV)

$GIO SET CCV port-address (4402 A000 440C, G$) C$()

The CCV defined by the array C$() is set (loaded) into the 2236MXE/Option W when the statement is executed. Here, port-address is the address of the TC port, and G$ represents the error/status/general-purpose registers.

Note: *The TC port's transmit and receive buffers, as well as the CSV and code translation tables, are cleared automatically when the CCV is loaded.*

Read Communications Status Vector (CSV)

$GIO READ CSV port-address (4403 1020 02FF 03FF 1223 C620, G$) A$

The information currently in the CSV 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 CSV are cleared automatically after the CSV information is read into the CPU. Insert a line containing a $BREAK statement before each line containing a Read CSV statement.*

Load Transmit Code Translation Table

$GIO LOAD TTBL port-address (4404 A000 440C, G$) C1$()

The transmit code translation table is loaded into the 2236MXE/Option W from an array if that array is previously defined in the application program. C1$() represents the array in the sample instruction above. The optional transmit code translation feature is enabled only if a transmit code translation table is loaded after the CCV is loaded into the 2236MXE/Option W.

The transmit code translation table should be exactly 256 bytes in length and represent the codes to which ASCII characters are to be converted prior to storage in the transmit buffer. The byte positions in the table should contain the *After Translation Characters* arranged in a sequence corresponding to the *Before Translation Characters*.

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 Chapter 1.

If no transmit code translation table is loaded by the application program, the standard transmit code translation table of the 2236MXE/Option W is used. The byte positions of the standard table contain the ASCII characters arranged in their before translation sequence, so that the characters are transmitted without change.

Load Receive Code Translation Table

$GIO LOAD RTBL port-address (4405 A000 440C, G$) C2$()

The receive code translation table is loaded into the 2236MXE/Option W from an array if that array is previously defined in the application program. C2$() represents the array in the sample instruction above. The optional receive code translation feature is enabled only if a receive code translation table is loaded after the CCV is loaded into the 2236MXE/Option W. (See the example in the section entitled "A Sample Program," lines 140-210 and 440.)

The receive code translation table should be exactly 256 bytes long. The byte positions in the table should contain ASCII characters (the *After Translation Characters* in this case) arranged in a sequence corresponding to the *Before Translation Characters*. The translation procedure is equivalent to using the binary equivalent of an incoming character's hexadecimal code as an index for a table look-up 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; the corresponding ASCII character should be located in the 25th position of the receive translation table. (Keep in mind that the first position in the table corresponds to the binary value zero.)

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

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).

If no receive code translation table is loaded by the application program, the standard receive code translation table of the 2236MXE/Option W is used. The byte positions of the standard table contain the ASCII characters arranged in their before translation sequence, so that the characters are received without change.

Disconnect

$GIO DISCONNECT port-address (4406, G$)

The 2236MXE/Option W disconnects from the line by setting the Data Terminal Ready signal to zero for a period of 3 to 5 seconds.

Send Break

$GIO BREAK port-address (4407, G$)

The 2236MXE/Option W sends a break signal if indicated by the low-order hexdigit in byte position 2 of the communications control vector. See Table 2-1 and Chapter 1.

Start Receiving Data (Start RCV)

$GIO START RCV port-address (4408, G$)

One *Start Receiving Data* statement is needed to enable data reception via the 2236MXE/Option W. If set for half-duplex mode, the transmit and receive buffers are cleared first. (These buffers are not cleared in full-duplex mode.) In both full-duplex and half-duplex mode, the receive timeout countdown is started by initializing byte 6 of the CSV to the value specified as the timeout interval (if different from binary zero). The 2236MXE/Option W then enters the receive mode and starts receiving data. After code translation (if enabled), the received data is stored in the TC port's receive buffer.

Transfer Received Data to the CPU (RCV)

$GIO RCV port-address (4409 1020 02FF 03FF 1223 C620, G$) D$()

All or part (if an end-of-record character is detected) of the receive buffer characters are transferred from the 2236MXE/Option W to the CPU and stored in the array D$(). (See Chapter 1.) The $GIO data buffer, denoted here by D$(), should be at least 256 bytes long since the 2236MXE/Option W has a 256-byte receive buffer associated with each port. 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.

Insert a line containing a $BREAK statement before each line containing a Transfer Received Data statement. The $BREAK statement causes execution of the "transfer received data" statement to start at the beginning of the next unit of processing time allotted by the BASIC-2 operating system to the partition where the program is running. This permits the whole data transfer to take place within the same unit of processing time.

Send Data (Send)

$GIO SEND port-address (440A A000 440C, G$) STR(F$(), 1, N)

If set for half-duplex mode, the receive buffer is cleared first. For half-duplex or full-duplex mode, bytes 1 through N of the array F$() are transferred from the CPU to the 2236MXE/Option W where they are stored in the transmit buffer after code translation (if enabled) and then transmitted.

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 2236MXE/Option W). The $GIO syntax requires a single argument format for arg-3. Therefore, a $PACK statement should be used to pack multi-argument data into a single argument prior to executing the *Send Data* $GIO statement if the application requires a specially formatted buffer.

Send Then Receive Data (Send RCV)

$GIO SEND RCV port-address (440B A000 440C, G$) STR(F$(), E)

This statement is applicable only for the half-duplex mode of operation and provides for automatic line turnaround. Beginning with the *Eth* byte, all remaining bytes of the array F$() are transferred from the CPU to the 2236MXE/Option W for storage in the transmit buffer after code translation is performed, if enabled. The 2236MXE/Option W transmits the data and then executes a *Start Receiving Data* operation.

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, and then the *Transfer Received Data* to CPU should be used. Use of the *Send Then Receive Data* $GIO operation automatically implements a line turnaround procedure, thereby ensuring the 2236MXE/Option W's readiness to receive data without loss.

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, in full-duplex mode, the 2236MXE/Option W remains in transmit and receive mode simultaneously and line turnaround does not occur.

Stop Transmitting (Stop Send)

$GIO STOP SEND port-address (440C, G$)

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 (Continue Send)

$GIO CONTINUE SEND port-address (440D, G$)

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. Use of this statement when transmission is already in progress does not harm the data.

Reset Controller

$GIO RESET MXE PORT port-address (4580)

This statement is used to unlock the 2236MXE/Option W. Locking can occur when the CPU tries to send bytes to a port for transmission when the port cannot transmit because of the state of the RS-232-C/V.24 signals. After this statement is used, the CCV should be reloaded. This statement need not be used before the initial loading of the CCV.

Set Signals

$GIO SET SIGNALS port address (440F A000 440C, G$) STR (A$, 1, 3)

or

$GIO SET SIGNALS port address (440F A000 4400, G$) STR (A$, 1, 3)

This statement is used to set or reset the state of RTS and DTR and optionally to lock these signals in a particular state.
Signal Sequences:

```
                          (0C)
CBS(0F) OBS(xxyyzz) CBS
                          (00)
```

where:

xx = a byte in which the '10' bit specifies the state of DTR and the '20' bit specifies the state of RTS

yy = a byte designating which bits in xx are valid

zz = a byte in which the '10' bit specifies the state of the DTR lock and the '20' bit specifies the state of the RTS lock

Byte xx designates the state of DTR and RTS. A bit value of one is the active state.

Byte yy is a mask that designates which bits in xx are valid. The '10' bit and '20' bit of yy correspond to the '10' bit and '20' bit of xx.

Byte zz allows the option to lock these states so that no condition can change them except another Set Signals command, a Reset command (CBS (80)), or a Disconnect command. The '10' bit locks or unlocks the state of DTR, and the '20' bit locks or unlocks the state of RTS. Byte zz is not affected by the values of bytes xx and yy. It is therefore possible to set one or both signals and/or lock these signals in any combination.
This command sets DTR and RTS to the states desired no matter what flow control option you select. Loading the communications control vector clears the lock states and sets DTR and RTS to their default values.

The signal states on the RS-232 connector are not changed until the CBS(0C) is received by the MXE controller. It is therefore possible to lock the current state of DTR and RTS without setting new signal states by writing three bytes to the MXE controller, with the third byte containing the lock state, but issuing a CBS(00) instead of a CBS(0C) to end the command.

The Disconnect command works as it currently does by setting DTR to inactive for 3 to 5 seconds and then setting it to active. The current lock state of RTS and DTR does not affect the Disconnect command. All other commands that alter the state of RTS or DTR first check the signal's lock state before changing it. If the signal is locked to its current state, the signal is not modified until a new command to unlock it is sent.

You need not send all three bytes to the MXE controller each time you want to change a signal. The three bytes (xxyyzz) are saved by the MXE controller and are overwritten one by one each time the command is issued. Thus you can overwrite zero, one, two, or three bytes, as desired.

Examples:

These examples give signal sequences and sample BASIC-2 code, using the $GIO statement, for those sequences. For more information on the $GIO statement, refer to the Wang *BASIC-2 Language Reference Manual*.

Example 1 sets RTS, sets DTR, and locks neither:

Signal Sequence

```
CBS(0F)   OBS(303000)   CBS(0C)
```

BASIC-2 Code

```
A$ = Hex(303000)
$GIO#1(440F A000 440C)A$
```

Example 2 resets RTS, sets DTR, and locks neither:

Signal Sequence

```
CBS(0F)   OBS(103000)   CBS(0C)
```

BASIC-2 Code

```
A$ = Hex(103000)

$GIO#1(440F A000 440C)A$
```

Example 3 does not change RTS; it sets DTR and locks RTS:

Signal Sequence

```
CBS(0F)   OBS(101020)   CBS(0C)
```

BASIC-2 Code

```
A$ = Hex(101020)
$GIO#1(440F A000 440C)A$
```

Example 4, by using CBS(00) instead of CBS(0C), changes neither signal but locks them both:

Signal Sequence

```
CBS(0F)   OBS(303030)   CBS(00)
```

BASIC-2 Code

```
A$ = Hex(303030)
$GIO#1(440F A000 440C)A$
```

A SAMPLE PROGRAM

The program listed in this section illustrates how a Wang system equipped with a 2236MXE/Option W terminal processor 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 seven data bits per character, even parity, and one 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 RETURN and DC1 (X-ON) characters are defined as terminators in the receive code translation table. Each RETURN HEX(0D) is translated to HEX(8D) prior to storage in the receive buffer. Each DC1 or X-ON character is translated to HEX(A0) prior to storage in the receive buffer. Upon transfer to the CPU, the high-order eighth bit of these end-of-record characters is changed from 1 to 0. Hence, each HEX(8D) becomes HEX(0D) which is the ASCII code for a RETURN, and HEX(A0) becomes HEX(20), which is the ASCII code for a space character.

6. Characters received with a parity or framing error are replaced by the substitute character HEX(7F), the ASCII code for a DEL character. Via the receive code translation table, each HEX(7F) is converted to a null character, HEX(00).

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

1. If the 2236MXE/Option W port used is not 03, 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 Chapter 1.

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.

BASIC-2 Code For The Sample Program

```
10 REM EXAMPLE --TTY EMULATION-- KYBD FOR INPUT, CRT FOR OUTPUT
20  DIM C2$(16)16, L$(255)1,  K$1, X$(20)1, Z$(7)1, P$3
21 REM C2$() 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$() IS A 20-BYTE COMMUNICATIONS CONTROL VECTOR
25 REM Z$() IS A 7-BYTE CPU ARRAY FOR READING STATUS VECTOR
26 REM P$ IS A 3-BYTE TC PORT ADDRESS
30 REM .....INITIALIZATION MODULE BEGINS
40 REM .,DEFINE $GIO MICROCOMMANDS TO OPERATE CONTROLLER
50    G0$=HEX(4402A000440C)       :REM  SET CONTROL VECTOR
60    G1$=HEX(4403102002FF03FF1223C620)      :REM  READ STATUS VECTOR
70    G3$=HEX(4405A000440C)       :REM  LOAD RCV TRANSLATE TABLE
```

```
80   G6$=HEX(4408)                :REM  START RECEIVING DATA
90   G7$=HEX(4409102002FF03FF1223C620)      :REM  TRANSFER RECEIVED
                                                  DATA
100  G9$=HEX(440BA000440C)        :REM  SEND-THEN-RECEIVE DATA
110  P$="A03":LINPUT "ENTER PORT # OF MXE PORT TO BE USED AS TC
           PORT", P$
112 SELECT TC  P$ : ERROR GOTO 110    :REM  SELECT TC PORT
114 SELECT #1 <P$>
120 REM ..DEFINE RCV TRANSLATION TABLE
130  INIT(00)C2$()           :REM  CLEAR RCV TRANSLATION TABLE
140  C2$(1)=HEX(000102030405060708090000B0C8D0E0F)  :REM  00-0F
150  C2$(2)=HEX(10A0121314151617181911A1B1C1D1E1F)  :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
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 2236MXE/OPTION W PORT
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$(3)>HEX(00)THEN 730          :REM  TEST FOR ERRORS
570    IF Z$(1)>HEX(00)THEN 740          :REM  TEST FOR BREAK
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>80 THEN 650                :REM  BRANCH IF A>80
620    $GIO /005(A000,A$)L$()<I,A>       :REM  DATA TO CRT IF A<81
630    D=D+A  :GOTO 690                  :REM  D IS OUTPUT POINTER
640 REM ..DISPLAY OVERRUN
650    B=81-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<81 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$;:GOTO 520
850    D=D-1:IF D>0 THEN 860:D=80:PRINT HEX (0C);
860    PRINT HEX(082008);:GOTO 520
870 REM ..BRANCH AS FOLLOWS FOR AN S.F. CODE HEX 07 THRU 0F
880    ON VAL(K$)-6 GOTO 840,850,800,840,520,520,520,520,890   :GOTO
                                                                520
890    PRINT "....SEND BREAK":$GIO #1(4407,A$)
900    GOTO 540
910 REM ...END OF KEYBOARD LOGIC
920 REM ....END OF MAIN LOOP
```

Wang's Asynchronous 1 (ASC1) software package, Release 7.0 or greater, which is supported by the 2236MXE/Option W terminal processor, has many features not illustrated in the sample program. With ASC1, data transmission and reception can be controlled over point-to-point, leased, or dial-up communication links between Wang systems and host computer systems that support TTY and 2741 line protocols. From the viewpoint of a Wang system operating under control of the TTY or 2741 emulation program of ASC1, 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, and received data can be output to a printer or disk. 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 you choose a set of communication 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.

If interested in additional information regarding use of the Wang ACS1 software package with the 2236MXE/Option W terminal processor, contact the Wang Sales Office in the area where a Wang system is being used.

APPENDIX A
ASCII CODE SET

The 2200 System character set is defined by 8-bit codes. The eight bits in each code are designated as high-order and low-order, with the high-order bits numbered 8, 7, 6, and 5 and the low-order bits numbered 4, 3, 2, and 1. Bit number 8 is 0, and the bits 7 through 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 CRTs 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 that accompanies a particular peripheral.

Table A-1. ASCII Code[a]

High-order 4 bits \ Low-order 4 bits hex-digit		$0_{0}0_{0}0_{0}0_{0}$	$0_{0}0_{0}0_{0}1_{1}$	$0_{0}0_{0}1_{0}0_{}$	$0_{0}0_{0}1_{1}1_{}$	$0_{0}1_{0}0_{0}0_{}$	$0_{0}1_{0}0_{1}1_{}$	$0_{0}1_{0}1_{1}0_{}$	$0_{0}1_{0}1_{1}1_{}$	$0_{0}1_{1}0_{0}0_{}$	$0_{0}1_{1}0_{0}1_{}$	$0_{0}1_{1}1_{0}1_{}$	$0_{0}1_{1}1_{1}0_{}$	$0_{0}1_{1}1_{1}1_{}$	$0_{1}0_{0}0_{0}0_{}$	$0_{1}0_{0}0_{0}1_{}$	$0_{1}0_{0}1_{0}0_{}$	$0_{1}0_{0}1_{1}1_{}$
	hex-digit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL 0	SOH 1	STX 2	ETX 3	EOT 4	ENQ 5	ACK 6	BEL 7	BS 8	HT 9	LF 10	VT 11	FF 12	CR 13	SO 14	SI 15
0001	1	DLE 16	DC1 17	DC2 18	DC3 19	DC4 20	NAK 21	SYN 22	ETB 23	CAN 24	EM 25	SUB 26	ESC 27	FS 28	GS 29	RS 30	US 31
0010	2	Space 32	! 33	" 34	# 35	$ 36	% 37	& 38	(apos.) ' 39	(40	) 41	* 42	+ 43	(comma) , 44	(dash) - 45	(period) . 46	/ 47
0011	3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
0100	4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
0101	5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	↑ 94	(underline) _ 95
0110	6	grave accent ` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
0111	7	p 112	q 113	r 114	s 115	t 118	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	¦ 124	} 125	~ 126	DEL 127

[a] 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
PORT ADDRESS DESIGNATIONS

Both Wang Model 2236MXE Terminal Processors and Wang Model 2236MXD Terminal Multiplexers can be used for communications between Wang terminals and a CPU; however, the 2236MXD cannot serve as an asynchronous communications controller and its use is not recommended with the CS. This appendix describes 2236MXD and 2236MXE port address designations.

A CPU can accommodate up to four 2236MXE terminal processors and 2236MXD terminal multiplexers, in any combination (see Table B-1). The first MXE or MXD supports terminals 1 through 4, and if an MXE, TC ports A01 through A04, as well. (Port A01 is not addressable, however, because it is reserved for the local system terminal.) The second MXE or MXD supports terminals 5 through 8, and if an MXE, TC ports A05 through A08. The third MXE or MXD supports terminals 9 through 12, and if an MXE, TC ports A09 through A12. The fourth MXE or MXD supports terminals 13-16, and if an MXE, TC ports A13 through A16.

Thus, if a 2236MXE is installed as the second terminal interface device on the central processor, port 1 of the 2236MXE can have only system terminal 5 and port address A05 designated for it. Similarly, if MXE terminal interface devices are installed as the first and third such devices and an MXD as the second, then port addresses A05 through A08 are invalid designations.

Since the 2236MXE can be used as both a terminal interface device and an asynchronous communications controller, the ports A02 through A16 associated with a 2236MXE can be configured as either terminal or TC ports.

Since the 2236MXD can be used only as a terminal interface device, the ports associated with a 2236MXD can be configured only as terminal ports.

Table B-1. Port Address Designations

Controller Number of 2236MXE or 2236MXD	2236MXE or 2236MXD Port Number	CS, MicroVP 2200MVP, or 2200LVP Terminal Number	2236 MXE Port Address[a]
1	1	1	Not Addressable
1	2	2	A02
1	3	3	A03
1	4	4	A04
2	1	5	A05
2	2	6	A06
2	3	7	A07
2	4	8	A08
3	1	9	A09
3	2	10	A10
3	3	11	A11
3	4	12	A12
4	1	13	A13
4	2	14	A14
4	3	15	A15
4	4	16	A16

[a]2236MXD TC ports are invalid.

APPENDIX C
SPECIFICATIONS

Power Requirements

The power is supplied by the CPU.

Interface Connections

The Interface Connections for the 2236MXE Terminal Processor are as follows:

Four 25-pin, EIA RS-232-C/CCITT V.24-compatible female plugs, each requiring a standard 12-foot (3.6 meter) Wang terminal cable to attach to a 2236D or 2336D terminal or a standard 12-foot (3.6 meter) Wang TC cable to attach to a DCE device or null modem

The Interface Connections for the Option W Terminal Processor are as follows:

Two 25-pin, EIA RS-232-C/CCITT V.24-compatible female plugs, each requiring a standard 12-foot (3.6 meter) Wang terminal cable to attach to a 2236D or 2336D terminal or a standard 12-foot (3.6 meter) Wang TC cable to attach to a DCE device or null modem

Asynchronous Transmission Rates

The asynchronous transmission rates are 50, 75, 100, 110, 134.5, 150, 200, 300, 600, 1200, 2400, 4800, or 9600\bits per second (bps).

Character Format Options

The character format options are as follows:

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

Communication Mode

The Communication Mode is full- or half-duplex.

Compatible Modems

Use the Wang Telemodem, Wang WA3451, or equivalent for full-duplex communications at 0-1200 bps. You can also use other full-duplex or half-duplex, Wang-compatible modems from other vendors.

Use the Null modem, available from Wang, for a direct communications link.

APPENDIX D
BASIC-2 STATEMENTS FOR MXE TC

SELECT TC

Format:

SELECT TC port-number

where:

port-number = {/Add | <alpha-variable>}

dd = 02-16

alpha-variable = 3-byte variable containing the port number ADD

The SELECT TC statement selects the specified port as a telecommunications (TC) port instead of a terminal port. No partition, including the partition issuing the SELECT TC statement, should be attached or assigned to the specified port when the SELECT TC statement is executed. If a partition is assigned or attached to the port, the system generates an illegal device error (P48). Therefore, you should release all partitions assigned to the requested port by issuing $RELEASE PART statements before issuing a SELECT TC statement or configure the system with no partitions assigned to the TC port.

The device address /Add or <alpha-variable> indicates which port to select. Port 01 cannot be selected as a TC port since Port 01 serves as the terminal port for the system console. If the specified port is selected as a TC port already, the system does not generate an error.

To avoid losing control of a port while selecting it as a TC port, perform the following steps. This example assumes that A$ is set to the device address.

```
$BREAK
SELECT #1  <A$>
$RELEASE PART
SELECT TC <A$>
$OPEN #1
```

The port designated by A$ should be attached to the partition initially.

Examples of valid syntax:

```
SELECT TC /A03
SELECT TC /A12
SELECT TC <A$>
```

SELECT TERMINAL

Format:

```
SELECT TERMINAL port-number
```

where:

port-number = { /Add, where dd is a decimal value from 02 to 16.
              { <alpha-variable>, where alpha-variable contains the port address Add.

The SELECT TERMINAL statement selects a port as a terminal port or converts a TC port to a terminal port. The specified port should not be opened (by means of $OPEN statement) as a TC port when executing a SELECT TERMINAL statement. If the specified port has been opened as a TC port by another partition, the system generates an illegal device error. When the system is powered on, the system selects all ports as terminal ports.

Examples of valid syntax:

```
SELECT TERMINAL /A04
SELECT TERMINAL <B$>
```

$OPEN

Format:

```
$OPEN [line-number,]  { device-address }  [, { device-address }]...
                      { file-number    }      { file-number    }
                      { port-number    }      { port-number    }
```

where:

device-address = /taa, where t is the device type and aa is the physical device address.

file-number = #n, where n is integer or numeric variable with a value 0 to 15, inclusive.

port-number = /Add, where dd is decimal value from 02 to 16.

The $OPEN statement requests exclusive use of a device (i.e., peripherals) or TC port for the current partition. To request exclusive use of a device, you specify either the device address or the file number in the $OPEN statement. To request exclusive use of a TC port, you specify the port number in the $OPEN statement.

The system signals an ERR P48 - Illegal Device Specification if one of the following conditions occurs:

- The specified device is not in the Master Device Table.
- The specified device is already opened for exclusive use by another partition.
- The specified port was not selected as a TC port before executing the $OPEN statement.

Once open, the current partition maintains exclusive control of the device or port until one of the following conditions occurs:

- Execution of a $CLOSE statement
- Execution of a program END statement
- Execution of a CLEAR or LOAD RUN command
- Execution of a SELECT TERMINAL statement
- Pressing of the RESET key

If another partition has already opened the specified device or TC port, the $OPEN statement branches to the specified line number. If no line number is specified, the $OPEN statement waits until the specified device or TC port becomes available for this partition.

If the programmer specifies multiple devices (including TC ports) in a $OPEN statement and the system cannot open one of the devices, then $OPEN does not open any of the devices.
Examples of valid syntax:

```
$OPEN /A13, /215, /A07
$OPEN /A02, #4
```

$CLOSE

Format:

```
$CLOSE [ { device-address } [ , { device-address } ] ... ]
       [ { file-number   }       { file-number   }       ]
       [ { port-number   }       { port-number   }       ]
```

where:

device-address	=	/taa, where t is the device type and aa is the physical device address.
file-number	=	#n, where n is integer or numeric variable with a value 0 to 15, inclusive.
port-number	=	/Add, where dd is decimal value from 02 to 16.

The $CLOSE statement releases the specified devices and TC ports that are currently restricted by the $OPEN statement. If no device address or port number is specified, the $CLOSE statement releases all devices and TC ports opened for the current partition.

The specified port must be selected as a TC port previously, otherwise the system generates an illegal device error. If the TC port has not been opened previously, the system does not generate an error.

Once open, the current partition maintains exclusive control of the port until one of the following conditions occurs:

- Execution of a $CLOSE statement
- Execution of a program END statement
- Execution of a CLEAR or LOAD RUN command
- Execution of a SELECT TERMINAL statement
- Pressing of the RESET key

Examples of valid syntax:

```
$CLOSE
$CLOSE /A15, #4
```

Format:

```
$GIO [comment] [ { device-address } [,] ] (arg-1 [,arg-2]) [arg-3 [; arg-3]...]
               [ { file-number   }      ]
               [ { port-number   }      ]
```

where:

file-number = #n, where n is integer or numeric variable with a value 0 to 15, inclusive.

port-number = /Add, where dd is decimal value from 02 to 16.

The specified port must be selected as a TC port previously, otherwise the system generates an illegal device error. If the specified port is opened by another partition, the $GIO statement waits until the specified port is available.

During $GIO through a TC port, the device address change commands, 71hh and 74r0, are not allowed.

For a description of the $GIO commands for asynchronous communication, See Chapter 2.

INDEX

T

V

W

WANG

Customer Comment Form

Publication Number 700-8098A

Title CS ASYNCHRONOUS COMMUNICATIONS USER GUIDE

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.

Printed in U.S.A. 14-3140 7-83

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.
PUBLICATIONS DEVELOPMENT
ONE INDUSTRIAL AVENUE
LOWELL,MASSACHUSETTS 01851

Fold

Cut along dotted line.