

M E M O R A N D U M

TO: 2200 TELECOMMUNICATIONS FILE
FR: TYLER B. OLSEN

DATE: 20 NOVEMBER 1974

RE: SPECIFICATIONS FOR BUFFERED ASYNCHRONOUS T. C. BOARD

INTRODUCTION

ASYNCHRONOUS WANG SYSTEM 2200 TELECOMMUNICATIONS THROUGH THE WANG 2227 BOARD HAS SEVERAL SHORTCOMINGS, PRIMARILY BECAUSE THE 2200 DOES NOT HAVE INTERRUPT CAPABILITY. LACKING INTERRUPTS IT IS NOT POSSIBLE TO ACCEPT LONG STREAMS OF DATA INTO THE 2200 WITHOUT DEDICATING THE MICRO-PROCESSOR TO A SPECIFIC DEVICE. AN ALTERNATIVE IS TO PROVIDE A LARGE BUFFER ON THOSE DEVICES WHICH HAVE A DATA STREAM THAT CAN NOT BE 'TURNED OFF'. SUCH OCCURS WITH AN INCOMING TELECOMMUNICATIONS DATA STREAM, WHERE THE 2200 IS BEING USED AS A TERMINAL WITH LITTLE OR NO CONTROL OVER THE INCOMING DATA.

WANG 2227 BOARD

THE WANG 2227 BOARD HAS A SERIOUS SHORTCOMING NOT ALLOWING THE SYSTEM 2200 TO CAPTURE DATA 'ON THE FLY' TO PERIPHERAL DEVICES. CURRENTLY, WE ADVISE GETTING BLOCKS OF DATA INTO 2200 MEMORY. BETWEEN BLOCKS, WE CAN OPTIONALLY SAVE THE DATA RECEIVED TO PERIPHERAL DEVICES.

- 1). THE 2227 BOARD HAS DOUBLE BUFFERING ON INPUT AND OUTPUT. THIS MEANS THE SYSTEM 2200 HAS TO BE READY TO RECEIVE DATA ON AN AVERAGE OF EVERY 9 MILLISECONDS AT 1200 BAUD/ 30 MS AT 300 BAUD.
- 2). DATA CAN BE TRANSMITTED FROM A SYSTEM 2200 QUITE EFFICIENTLY VIA PRINT OR PRINTUSING STATEMENTS.
- 3). IN THEORY, DATA CAN BE RECEIVED BY A SYSTEM 2200 USING ONE OF THE FOLLOWING COMMANDS.
 - A). INPUT (RESTRICTED TO STRING OR NUMERIC VARIABLES).
 - B). A KEYIN LOOP (RESTRICTED TO 300 BAUD IN PRACTICE).
 - C). DATALOAD BT (CANNOT BE USED BECAUSE THE MICRO-CODE HAS NO ERROR DETECTION, IE. SPECIAL FUNCTION HANDLING.)
 - D). \$GIO (SPECIAL COMMAND FOR OPTION 2)
- 4). PROGRAMMING THE 2200 TO ACCEPT DATA 'ON THE FLY' IS DIFFICULT AND REQUIRES TIGHT CODING AND/OR PADDING OF DATA STREAMS.

LINE BUFFERED ASYNCHRONOUS T. C. BOARD

SPECIAL BUFFERING OF INCOMING DATA CHARACTERS IN A CONTROLLER BOARD WOULD ALLOW SAVING OF DATA 'ON THE FLY' TO A VARIETY OF 2200 PERIPHERALS INCLUDING CASSETTE, DISK, OR PRINTER.

RELATIVE SPEEDS--

- 1). SPEED OF WANG 2200 PERIPHERALS--(INCLUDES 150 MILLISECOND OVERHEAD PAD)
 - A). 2217 CASSETTE--950 MILLISECONDS TO WRITE A 256 BYTE CASSETTE RECORD.
 - B). 223X DISK --158 MILLISECONDS TO WRITE A 256 BYTE SECTOR.
 - C). 224X DISK --171 MILLISECONDS TO WRITE A 256 BYTE SECTOR.
 - D). 2261 PRINTER AT 125 LINES PER MINUTE REQUIRES .4 SECONDS PER LINE.
 - E). 2231 PRINTER REQUIRES 112 TO 680 MILLISECONDS PER LINE.
- 2). SPEED OF DATA BLOCK RECEPTION--(IN MILLISECONDS)

BAUD	128 BYTES	256 BYTES
300	3840	7680
1200	1152	2304
1800	768	1536

A BUFFERED INPUT STREAM WOULD ALLOW 'ON THE FLY' OUTPUT TO A VARIETY OF WANG PERIPHERAL DEVICES.

PAGE 2 BUFFERED ASYNC. BOARD
DESIRABLE FEATURES

- 1). ERROR DETECTION AND NOTATION
 - * PARITY ERRORS--DENOTE WITH SPECIAL FUNCTION FLAG
 - UNCLX PROGRAM CONTROL ALLOW CHOICE OF CHARACTER IDENTITY.
 - 1. CHARACTER RECEIVED.
 - 2. SPECIAL CHARACTER.
 - 3. CANNED CHARACTER.
 - * FRAMING ERROR--IBID
- 2). BREAK
 - BREAK--BE ABLE TO RECEIVE/RECOGNIZE UNDER PROGRAM CONTROL
 - BREAK--BE ABLE TO SEND EITHER MANUALLY OR UNDER PROGRAM CONTROL
- 3). MODEM SUPPORT--
 - * 1). BELL 103-A3 UP TO 300 BAUD.
 - * 2). BELL 202-C UP TO 1200 BAUD.
 - 3). BELL 202-T UP TO 1800 BAUD.
- 4). BAUD RATE SUPPORT--
 - * MANUAL SELECTABLE--PROGRAM SELECTABLE
 - * 110 BAUD STANDARD TTY (ASR33/35)
 - * 134.5 BAUD IBM 2741 SPEED
 - * 150 BAUD
 - * 300 BAUD (30 CPS TTY COMPATIBLE DEVICES).
 - * 600 BAUD
 - * 1200 BAUD 120 CPS DEVICES
 - * 1800 BAUD
 - 2000 BAUD QUESTIONABLE NEED
 - DEC ALSO SUPPORTS 0, 50, 75, 200, 2400, 4800, 9600 BAUD.
- 5). CHARACTER BIT CONFIGURATION--
 - * SWITCH SELECTABLE PROGRAM SELECTABLE
 - * DATA BITS 5, 6, 7, OR 8
 - * PARITY OR NOT IF YES, ODD OR EVEN
 - * STOP BITS 1, OR 2 (PERHAPS ADD 1.5).
 - 2227 BOARD SUPPORTS 1.5 STOP BITS WITH 5 BIT CHAR.
- 6). HALF DUPLEX/FULL DUPLEX--PROGRAM SELECTABLE.
IF HALF DUPLEX, RECEIVER IS BLINDED DURING TRANSMISSION OF A CHARACTER.
- 7). AUTO-ECHO ENABLE--POSSIBLE WITH \$GIO COMMAND
- 8). SET UP A CIRCULAR DATA BUFFER. (FIFO)
 - BUFFERING ON INPUT (REQUIRED) ON OUTPUT (OPTIONAL)
 - A). 128 BYTE=
 - B). 256 BYTE=
 - C). SIMULTANEOUS SEND/RECEIVE?RECEIVER AND TRANSMITTER EACH CONTAIN A DOUBLE BUFFER CONSISTING OF:
CHARACTER HOLDING REGISTER + SHIFT REGISTER
- 9). ALLOW BUFFER EMPTYING ON:
 - A). CHARACTER BASIS.
 - B). LINE BASIS (HOW DEFINE LINE?)
 - C). TIME OUT? (ON CHIP?)
- 10). PREPARE SPECIFICATIONS WITH JUSTIFYING LOOSE LEAF APPENDIX
PROGRAM INTERFACE IS SET-UP WELL IN PDP DH 11-16 MANUAL.
- 11). AN MCS 4 CHIP OR MCS 4040 CHIP + PROM + ROM + RAM
SHOULD BE CONSIDERED FOR IMPLEMENTATION.
- 12). SELF-CHECKING: EACH WANG SERVICEMAN SHALL BE PROVIDED WITH AN RS-232
COMPATIBLE PLUG WHICH SHALL LOOP THE OUTPUT TO THE INPUT SIGNAL.
- 13). REVERSE CHANNEL CAPABILITY--WHAT TO DO IS UNCLEAR.
- 14). ABILITY TO HANDLE TWX AND TELEX NETWORKS
THIS INVOLVES LOWER BAUD RATES AND EITHER CURRENT LOOPS OR HIGH
VOLTAGE LINES. L. MUNINI AND OTHERS SUGGEST AVOIDING PITFALLS.
- 15). EOM SWITCH (THIS 2227 FEATURE IS UNNECESSARY WHEN USING \$GIO OR A
KEYIN LOOP. IE. EOM CAN BE PROGRAMMED.

WANG

LABORATORIES, INC.

MEMORANDUM

TO: 2200 TELECOMMUNICATIONS FILE

FROM: Bob Kolk

DATE: January 14, 1975

SUBJECT: PRELIMINARY SPECIFICATIONS: 22X7 Buffered Asynchronous
Telecommunication Interface

I. INTRODUCTION

Over the past year it has become apparent that the present 2227 Telecommunication interface is not completely adequate in supporting asynchronous telecommunications. This document will attempt to identify major problem areas existing with the current 2227, and to specify a new interface that will hopefully overcome most of these. Along with the binary synchronous interface, it should provide a fairly complete telecommunications capability for the 2200 series.

II. CURRENT PROBLEMS WITH THE 2227

To better understand the important additional features in the 22X7, it is useful to first review the limitations in the 2227, and discuss how they can be corrected in the 22X7.

1. Buffering

The major problem that currently exists in the 2227 is the lack of line buffering. This problem is particularly critical when receiving data. In an application where an incoming stream of data lines must be printed or saved on disk or tape a general problem arises in the 2200; since certain discrete time periods must be taken by the 2200 to output the data. During this time, an incoming data line cannot be received by the 2200. Since the 2227 buffers only a single character, a relatively small amount of time, (1 character time), is available to complete output operations. For higher baud rates this is often insufficient. Therefore, the 22X7 spec proposes as a major feature, the ability to buffer both received and transmitted data lines. A minimum of 512 bytes of memory will be provided for this. There are several important requirements in the 22X7 for having a certain degree of intelligence (i.e., a microprocessor) in the 22X7. One is line buffering. If the 2200 can receive and send complete lines, the 2200 programming requirements are substantially simplified, and more time is available for storage and retrieval activities with peripheral devices. In conjunction with receiving lines of characters are associated problems such as identification of multi-line termination characters, error detection on a line-by-line basis, support of break conditions. All of these can be adequately supported, if the 22X7 controller has a certain degree of intelligence, (i.e., a microprocessor).

2. Baud Rates

The current 2227 supports 110, 150, 300, 600 and 1200 baud. 134.5 baud (for 2741 compatibility) must be added. In addition, it is probably important to consider 1800, 2400, 4800 and 9600 baud. Although higher baud rates are used less frequently in asynchronous transmission, if they can be provided for relatively inexpensively, they will provide a more complete capability. It is also possible that the 22X7 might provide an excellent vehicle by which relatively high speed RS-232-C compatible peripherals can be interfaced to the 2200, (such as graphic display terminals, special card readers, punches, printers). In this case the higher baud rates (i.e., 4800, 9600) and buffering could substantially improve operating performance. If the high baud rates were included for interface applications it may also be desirable to incorporate built-in "null-modem" pin switching capability possibly with a switch on the board.

3. Break Generation/Detection

Currently the 2227 cannot initiate breaks under program control. In addition incoming breaks are not distinguishable from parity/framing errors. In some applications break transmission and detection are relatively important and must be adequately supported. In the 22X7, it is proposed that transmission of a break signal be made programmable, (i.e., an interrupt/type command be sent from the 2200 to the controller microprocessor which can then send the signal). This would eliminate the necessity of a separate remote switch and provide greater flexibility. (A keyboard special function switch can be used). In addition, the presence of a microprocessor in the controller should also provide a more precise detection of an incoming break signal (i.e., time-out a framing error). Thus parity error, framing error, loss carrier and break can be discrete inputs to the 2200.

4. Half Duplex/Full Duplex/Reverse Channel Capability

Electronically, the capability of supporting half or full duplex operations is a function of the modem. 103 type modem provides separate and discrete frequencies for simultaneously sending and receiving information. (A function of who is the originator). Standard 202 type modems provide only one way (2-wire, half duplex) transmission with a 200 ms turn-around requirement. Options are available on the 202 to support a low speed "Reverse Channel", (0 - 5 bits a second). The current 2227 can only functionally support half duplex operations. (i.e., the 2200 can only be sending or receiving at a given instant). This is primarily because of the nature of the 2200 operations controlling the 2227. Break detection during output operations is not supported.

Simultaneous full duplex operation with entirely different data is not generally used in telecommunications. There are however, several applications where certain types of full duplex operations are significant and should be supported. They are:

1. The capability of sending or receiving a break signal (either via the 103 type alternate channel or the 202 normal or reverse channel) at any time including during time when data is being received or sent.
2. The capability of supporting echo-plex (which is often referred to as full duplex). This is when each transmitted character is echoed back for verification. (This would only be valid for 103 type operations).

3. Use of the reverse channel to detect level changes. Since the 202 reverse channel operates at such a low speed it is not generally used to transmit characters. It is however, used as a control level. An example of this is where a 200 ms level change on the reverse channel indicates break. A number of RS-232-C compatible peripheral devices also utilize the reverse channel as a level to indicate ready condition, such as "ready to print the next line" for a printer. Therefore, it is desirable for the 22X7 to be able to set and detect the reverse channel both for break and non-time dependent situations.

5. Multi-EOM Characters, Break Characters

Many time-sharing services use different end of message (i.e., end of line) characters, and some use several. In addition some devices use special characters to indicate break. In the 2227, one additional EOM character can be setup by switches and is decoded into a carriage return. In situations where several EOM characters were sent, this meant rather complex KEYIN loops had to be programmed to detect and separate lines and input characters had to be received one at a time by the 2200 program. To provide greater flexibility, the 22X7 will be able to detect up to 5 EOM/Break characters. They will be selectable via program control. (i.e., the 2200 program will initialize the controller by transmitting control information to it which allows the EOM characters to be detected. This also makes the received data relatively code independent since the ASCII carriage return is no longer a default EOM character.

6. Automatic Code Conversion Facility

In certain types of telecommunications, ASCII characters are not normally sent or received. An example of this is 2741 compatible telecommunications. Although code conversion can generally be handled via BASIC software, there are several reasons why it may be desirable to handle it in the 22X7 controller. They are:

1. To do code conversion at relatively high speed, a 2200B or 2200C with Option 2 (i.e., the \$TRAN statement) is required. This makes a relatively expensive system. Since in general, asynchronous telecommunications options in competitive equipment tends to be priced in a substantially lower price range than bisync, we may risk pricing ourself out of the market for telecommunications applications that require code conversion.

2. The implementation of code conversion algorithms in the controller board using microprocessor is not extremely difficult. The conversion tables can be loaded from the 2200. Whether or not code conversion is done has little implication on the hardware design, it would be an excellent sales feature.
3. Code conversion implemented in the 22X7 controller makes more time available in the 2200 CPU for peripheral device storage and printing. Without the \$STRAN statement code conversion could take about 20 ms per character. Code conversion for 2741 codes which require upshift/downshift logic cannot be handled by \$STRAN as efficiently and could take substantially more time per character.

7. TWX/TELEX Capability

Some forms of TWX and TELEX could be handled on the 22X7 board but others present several problems:

- (1) The A. T. & T. 84D4 full duplex protocol is complex and should not be supported.
- (2) Simpler half duplex protocols such as the 83B3 could probably be supported with BASIC software if proper baud rates are available.
- (3) The availability of the proper baud rate tends to be the most significant problem. There is not one standard baud rate. Various TWX/TELEX networks require baud rates such as 45, 55, 65, 70, etc. Some manufacturers have taken an approach where one baud rate selection is linked to a replaceable crystal clock, thus a relatively narrow range of TWX/TELEX baud rates can be supported on a modular basis. We should probably consider some approach similar to this. (i.e., crystal or special piggyback board).

8. Limitations of 2200 BASIC Statements, (INPUT, KEYIN)

When receiving data with the current 2227 board with a BASIC INPUT and/or KEYIN statement, there are several limitations:

1. INPUT

Messages are limited to 64 characters unless special provision is made to send messages in special formats. A number of ASCII characters such as a comma, carriage return, backspace, HEX(00), HEX(5C), HEX(7F) result in special interpretation. Finally, data received with INPUT is always echoed to the CRT (or device currently selected for Console Input). These restrictions tend to make the INPUT statement difficult to use for any telecommunications application which receive data. Therefore, we do not recommend or support the use of INPUT with the 22X7 board.

2. KEYIN

KEYIN is available on the 2200S or 2200B/C. It allows characters to be received one at a time, tested and/or stored in an alpha variable or array. A program loop using KEYIN takes a minimum of 10 - 20 milliseconds/characters. Since the implication of using the 22X7 board is that the buffer should provide additional 2200 CPU time to save or print an incoming data stream, this would restrict the use of KEYIN to 300 baud or less.

3. PRINT

PRINT is generally adequate for telecommunications output although it has some programming inconveniences. Output lines must be broken up into specific 64 character blocks. (i.e., alpha variables or array elements), and the program must strictly adhere to rules which avoid the output of automatic carriage returns. Also when an ASCII carriage return character is transmitted, a LF or null character is sent automatically after it.

4. Most Desirable Statements for use with the 22X7

To efficiently receive or send data via the 22X7 the following statements should be considered.

a. DATALOAD BT (Paper Tape or Teletype) -- (2200B/C, 2200S OP 22)

This statement allows any number of 8-bit codes to be received and stored in an alphanumeric array either specified by count or termination characters. Because it supports relatively high speed input and has no character code or line length restrictions it is highly suitable for telecommunications input and if supported would not require a customer to buy Option 2. The previous limitations with using this statement for the 2227, (i.e., no parity detection), are no longer valid, since the 22X7 will have a microprocessor and the parity, break, and line length information can be sent to the 2200 discretely prior to receiving the line.

b. DATASAVE BT (Teletype) -- (2200B/C, 2200S OP 22)

This statement is extremely useful for transmitting lines of any length and character code with no restrictions. This eliminates some of the inconveniences with using PRINT. Since it appears that probably a minimum system for the 22X7 should include DATALOAD BT, DATASAVE BT will also be present and should be the recommended output statement.

- c. Option 2 (\$GIO, \$IF ON, \$STRAN) -- (2200B/C with Option 2, 2200S OP 23)

If DATALOAD BT and DATASAVE BT are supported for the 22X7, Option 2 will not be absolutely required, which in many cases makes the asynchronous telecommunications capability less expensive. It will however, provide some particular advantages for certain applications. If code conversion is not supported in the 22X7, the \$STRAN statement will do it much more efficiently. For 2200 to 2200 applications an LRC can be sent and checked with each data block increasing reliability. Messages can be blocked and unblocked for efficiency.

9. Separate Selection of INPUT/OUTPUT, Data Format, Parity

There have been a few occasions with current 2227 customers, where transmitted and received parity requirements were different. This is probably not a typical situation, however, if separate transmitter/receiver chips are used, the data format/parity information for sending/receiving could be kept in separate microprocessor registers. A satisfactory alternative may be to simply ignore incoming parity errors which has no hardware implication on the 22X7.

10. Additional Features

With the availability of a microprocessor in the 22X7, there are additional features which could be supported. Monitoring and detection of the carrier is useful information for the 2200 control program, since it could be used to detect carrier drops and insure proper connections. It should probably be available on the initial version of the 22X7. Better provisions may now be available to support automatic dialing in conjunction with that modem equipment, although it is probably not necessary for the initial version of the 22X7.

11. Self-Checking Facility for the 22X7

One current deficiency in the 2227, is the inability to "self-check" the 2200/2227. A relatively simple approach would be a switch on the board which would echo transmitted data back to the receiver chip. If the setting of this switch were detectable by the microprocessor, the microprogram could then automatically receive the transmitted information, check it, and pass valid or error information back to the 2200. It is also sometimes useful to monitor certain signals on the RS-232-C connector when attempting to diagnose a telecommunications problem. They are:

BA	Transmitted Data
BB	Received Data
CA	Request to Send
CB	Clear to Send
CC	Data Set Ready
CD	Data Terminal Ready
CF	Carrier Detect
SA	Supervisory Transmit
SB	Supervisory Receive

If these signals can be monitored by LED indicators, it would be helpful if cost is not to great.

III. FUNCTIONAL SPECIFICATION FOR 22X7

The 22X7 option is a line buffered asynchronous telecommunications interface available on the 2200A, B, C or S. Recommended use is for a 2200B, 2200C or 2200S with OP 22. It allows the 2200 system to communicate via serial asynchronous transmission on leased or private lines at up to 2400 baud. It supports half duplex or full duplex operation, the latter generally providing break detection and echo-plex capability. It can also support the interface of RS-232-C compatible devices at up to 9600 baud.

The 22X7 replaces the 2227 option and interfaces with BELL 103A3, 202C, and 202D asynchronous modems.

The salient features of the 22X7 are:

1. Full Input/Output Line Buffering

Up to 512 characters of Input or Output data buffering. Integral lines can be received, buffered and transmitted to the 2200 when available. This allows the 2200 to PRINT or STORE incoming data, asynchronously.

2. Program Selectable Character Format

(5, 6, 7, or 8 Data bits; 1 or 2 Stop bits; Even, Odd or No Parity).

3. Program Selectable Baud Rates

(110, 134.5, 150, 300, 600, 1200, 1800, 2400 baud for telecommunications). (4800, 9600 baud for peripheral interface). Optionally, special clock selection for TWX/TELEX (45, 55, 65, 70 baud).

4. Program Selectable End of Message Characters

Up to 5 program selectable EOM characters can be specified. EOM characters can be sent separately to the 2200 for each line received for easy examination.

5. Program Selectable Full or Half Duplex Operations

For full duplex operations, incoming break, echo-plex and reverse channel level change is supported during transmission or receipt of data.

6. Program Selectable Supervisory (Reverse) Channel Operations

For 202 type modems with supervisory channel operations to send or receive break and/or change or detect the reverse channel level can be programmed.

7. Full Programmable Break Detection and Generation

Breaks can be transmitted or detected under program control for using either the standard carrier frequencies on the 103 or 202, or the supervisory channel on the 202. (The 2200 special function keys provide a convenient means by which break can be initialed via program control.

8. Complete Error and Status Detection Capability

With each line received, the 2200 will receive in separate error and status information consisting of:

- a. EOM characters which terminated the line.
- b. Number of characters in the line.
- c. Parity error.
- d. Framing error.
- e. Break received.
- f. Overrun.

Optionally, each character in a line which had a parity error can be delineated by setting the high order bit = 1.

In addition, the 22X7 can be scanned at anytime during transmitting or receiving operations to obtain the following status information.

- (a) Full or Partial input line in 22X7 buffer, number of characters, error characters, error conditions, etc.
- (b) Output line being transmitted or done, number of characters.
- (c) Break Received.
- (d) Carrier Level.
- (e) Supervisory Level.
- (f) Time-out Condition.
- (g) Other pertinent RS-232-C levels (Optionally ?).

9. Complete Asynchronous Timing Provision

The following time delays are introduced by the 22X7 to insure compatibility with a wide range of terminals and processors.

1. Delivery of the EOM characters and/or a complete line to the 2200 is delayed one character time to allow the transmitting terminal time to recover for reception.
2. Transmission of the first character following the rise of Clear to Send is delayed one character time to allow the receiving station to stabilize on carrier.
3. In half duplex mode the drop of request to send is delayed 2 msec. following the delivery to the modem of the stop mark of the last transmitted character to allow the modem sufficient time to modulate a complete stop time. The delay does not include the time required for the character to clear the VART.

10. Program Selectable Time-out

During input operations if so selected a time-out condition will be set if an EOM character has not been received for a 30 second period following the receipt of at least one message character. The condition can be detected by the 2200 via a normal status scan operation.

IV. HARDWARE SPECIFICATIONS AND CONSIDERATIONS

The 22X7 will generally consist of the following items:

- A microprocessor (LSI type)
- PROM chips for the controller microprogram
- A minimum of 512 bytes of RAM for data buffering. This will be accessed solely by the microprocessor
- Transmitter and Receiver Chip
- Sufficient microprocessor registers to hold all necessary format status levels to control the transmitter/receiver chip, RS-232-C levels required for scanning by the microprocessors and data buffers
- 2200 Bus address selection and data transfer logic

1. Microprocessor

A variety of microprocessors could be considered, MCS 4, MCS 4040, MCS 8008 or MCS 8080. There is generally no specific requirement for interrupt. Although the MCS 4 may be marginally adequate, its speed and instruction set limitations do tend to make it relatively inefficient and difficult to program. For example, it could take 20 - 30 milliseconds to send or receive a 100 character line to or from the 2200. On the other hand cost is more of a consideration in the 22X7 controller than for instance is bisync. A realistic target price range for the 22X7 is probably \$1000 - \$1500. An 8080 can be considered if we can still meet these goals. It may be worth considering the 8080, if getting to a higher volume level on the chip brings down the unit price.

2. PROM

The use of PROM for all microcode has both PROS and CONS. The advantages being:

- It simplifies operating procedures
- It simplifies programming procedures
- It makes 22X7 operation less prone to operational/programming bugs
- Since programming requirements for the 22X7 are not as extensive or subject to change as bisync, therefore, its more feasible

The disadvantages are:

- Somewhat higher cost. This however, depends largely upon what type of RAM is used.

3. RAM

It appears that 512 bytes of RAM is generally adequate for data buffering. If code conversion is supported in the 22X7 microprocessor, conversion tables would take up some of the available RAM. In the case of 2741 conversion, this would generally mean only 64 characters each for two tables. For other 8-bit codes it could take substantially more. It may therefore, be important to allow provisions in the board layout for possibly 768 or 1024 bytes if efficient code conversion is a consideration for the 22X7.

4. Transmitter/Receiver Chips

There are no specific requirements other than to meet the requirements of this specification as efficiently as possible and can easily be interfaced to microprocessor registers which control information to select data/parity format, baud rate, etc. Using separate transmitter/receiver chips such as those on the bisync board can reduce board logic for handling different baud rates. It may be beneficial to utilize chips currently used or planned for other products to increase volume and reduce unit price. The transmitting and receiving logic must be designed to operate independently and asynchronously. That is, simultaneous transmitting and receiving operations can go on at the same time under control of the microprocessor. This is necessary to support 103 type break and echo-plex operations.

5. Microprocessor/Transmitter/Receiver/RS-232-C Connector Interface

Since the microprocessor in the 22X7 board will have direct control over all transmitting and receiving operations sufficient registers and logic paths must be provided to support all requirements. This will generally consist of the following.

a. Format/Baud Rate Selection

Microprocessor register(s) will be semipermanently set to binary values which select the character format and baud rate as follows.

Word Length	-- Two bits	-- (5, 6, 7, 8 bits)
Parity	-- Two bits	-- (Even/Odd/Inhibit)
Baud Rate	-- Approximately 6 - 8 bits	-- (110, 134.5, 150, 300, 600, 1200, 1800 and 2400 baud) Plus TWX/TELEX baud rates either separate crystal or logic for 45, 55, 65 and 70 baud.

These can probably be shared by transmitter/receiver chips. Although there is some requirements for separate send/receive settings, it can probably be supported by simply ignoring parity error on incoming data.

b. Holding Register Interface

- . Register(s) to receive or send data between transmitter/receiver holding registers
- . Send/receive strobes
- . Two bits to examine data received and data sent flags

c. Receiver Errors

- Three bits -- (Overrun Error, Framing Error, Parity Error)

d. RS-232-C Signals

To support operation of breaks and reverse channel, the following RS-232-C signals must be either available in a microprocessor register or sent from a microprocessor register.

Signals monitored by microprocessor

<u>PIN</u>	<u>FUNCTIONS</u>
3	Received Data (To monitor break)
5	Clear to Send (202 convention)
6	Data Set Ready (overall control)
8	Carrier Detection (overall control)
12	Supervisor Channel Received Data (To monitor)

Signals set by microprocessor

<u>PIN</u>	<u>FUNCTIONS</u>
2	Transmitted Data (To override transmitter chips to send break)
4	Request to Send (202 convention)
11	Supervisor Channel Transmitted Data (Send break, etc.)
20	Data Terminal Ready

6. Break Generation/Detection

- Break signals will be sent upon request from the 2200 by the microprocessor as follows:

103/202 -- Hold Pin 2 (Transmitted Data) to an override mark condition for 200 ms via register setting.
202 Reverse -- Hold Pin 11 to an override mark condition for 200 Channel ms.

- Break signals will be received as follows:

103 -- During transmission or receiving operations. One character is received with a framing error and/or followed by 125 ms time period where no characters are received and Pin 3 (received data) is observed to be at a mark state.

202 -- During receiving operations only 1 framing error character and/or 125 ms. of Pin 3 are at mark state.

202 Reverse Channel -- During transmission operations, the supervisor input channel Pin 12 goes to a mark state and remains for 125 ms. (One shot logic may be required to insure a steady state signal).

7. Microprocessor/2200 I/O Interface

It appears highly desirable at this time to support four unique device addresses for the 22X7 board. To accomplish this the low order two bits of the device addresses available on the 2200 I/O bus will not be used for card enabling but will be latched into a microprocessor register on the 22X7 during the ABS strobe. These two bits along with a bit that indicates the enable state of the 22X7 will permit the microprocessor to perform the following four I/O functions:

<u>Low Order Address Bit</u>	<u>Function</u>
00	Send operational parameters or commands from 2200 to 22X7. Parameters -- Data Format, Baud Rate, EOM characters, etc. Commands -- Send Break, Receive a line, Send last line and turn-around, Receive a line and turn-around.
01	Scan 22X7 status. 22X7 sends 2200 a block of data indicating status of input or output operations, (i.e., line received, being received, count, line sent, being sent, count, break received, error conditions, RS-232-C conditions (Reverse Channel, Carrier, etc)).
10	Send next output line from 2200.
11	Receive next input line into 2200, or all available characters.

The input/output operations between the 2200 and 22X7 can then operate via the following registers and flags.

2200 → 22X7

- 22X7 sets a register bit to logic 0 to indicate Ready (\overline{RB}).
- 2200 strobes 8-bits into a microprocessor input register via an \overline{OBS} strobe which also sets the ready/busy bit to 1.

22X7 → 2200

- The 22X7 tests the 2200 \overline{CRB} bit via a register.
- When ready (logic 0) to send 8-bits of data from a microprocessor output register via a strobe gated to \overline{IBS} .
- Output strobes from 2200 are ignored and \overline{RB} generally is set to ready. (This can possibly be done by the microprocessor).

As per all 2200 controllers, all I/O logic levels are gated via the board enabled.

This is a slightly different approach that the bisync/2200 interface which utilizes the 2200 \overline{CBS} strobe. The major reason for the difference is that it is probably most desirable from a marketing point of view to not require 2200 Option 2 for the 22X7 operation. The added cost of Option 2 could make the 22X7 offering fairly expensive since competitive asynchronous telecommunications option are often substantially less expensive than bisync.

V. OPERATION SPECIFICATIONS

It appears desirable at this time to support the operation of the 22X7 primarily with the DATASAVE BT and DATALOAD BT statements currently used for the teletype/paper tape. There are four advantages including this:

1. It eliminates the many problems associated with using the INPUT verb, (i.e., 64 character line lengths, restrictions existing with certain characters such as CR, comma, etc., and often undesirable automatic echoing).
2. It provides the capability to receive and/or send any 8-bit character codes of any line length.
3. It eliminates the necessity of having to purchase Option 2 (for \$GIO) which increases the total cost of telecommunications.
4. Device addresses (which select the four address modes) can be listed in the statement. This simplifies programming.

There are four modes of operation for the 22X7. These are enabled by the use of four unique device addresses for the boards using the low-order two bits of the address for mode selection. The modes operate as follows:

1. T. C. Parameter Load Mode (Address XXXXXX00)

In this mode parameters are transmitted from the 2200 which primarily initialize the board for T. C. operation. The parameters are received by the 22X7 microprocessor, decoded, and saved in RAM or registers.

It is proposed that these parameters be sent in an "atom" type ASCII character format. Although this will require additional 22X7 microprocessor steps to decode it could be significantly easier for the typical user to program and understand.

When the "Parameter Load Mode" address enables the 22X7 board, one or more of these parameters can be sent. There are two types of parameters: "permanent" and "dynamic". "Permanent" parameters are information such as data format, baud rate, EOM characters, which are generally sent initially and left unchanged. "Dynamic" type parameters are dynamic commands, which cause some action in the 22X7, (i.e., send break, turn-around for input, etc.). Space characters are null parameter atoms (ignored).

Typical parameters are sent via the DATASAVE BT or PRINT statement.

```
100 A$ = "D7S2 P01 B1200"
110 DATASAVE BT /418, A$
```

```
200 SELECT PRINT /418
210 PRINT "D7S2 P01 B1200"
```

A. Permanent Parameters [] means optional

<u>Format</u>	<u>Function</u>
(1) Data Bits - <u>DX</u>	Number of Data Bits, where X = 5, 6, 7 or 8
(2) Stop Bits - <u>SX</u>	Number of Stop Bits, where X = 1 or 2
(3) Parity - <u>PXY[ZZ]</u>	X = Parity Selection X = N (No parity) X = E (Even) X = O (Odd)
	Y, ZZ = Parity Error Action: Y = 0 Indicate parity error by line only Y = 1 Set the high order bit of each char. with parity error =1

Y = 2 Replace each char. with parity error by ZZ
 Y = 3 Complement of parity error
 Y = 4 Ignore parity error
 ZZ = Two hex digits which replaces characters with parity error (If Y = 2)

(4) Baud Rate - BXXXX Baud Rate where XXXX = 0110

0134
 0150
 0300
 0600
 1200
 1800
 2400
 4800 (Interface Only)
 9600 (Interface Only)

and optionally if TWX/TELEX

0045
 0055
 0065
 0070

(5) End of Message Characters EXNYY [YY YY YY YY]

End of Message characters
 X = S strip EOM character when sending line to 2200
 L leave EOM character when sending line to 2200
 (Note, EOM character is normally sent on scan data)
 N = Number of EOM characters
 YY = EOM characters (2 hexdigit/character)

(6) Modem Selection MXXXX

Modem Selection XXXX = 0103 (103A3)
 = 0202 (202C)
 = R202 (202 with reverse channel)
 = F202 Full duplex 202 (4-wire)

2 character

(7) Trailing Message Character Omission

TTY

TT = Omit trailing characters to be transmitted in a line sent from the 2200

TRY

TR = Omit trailing characters to be received in a line by 2200 where YY are two hex digits indicating the code the trailing character to be omitted.

(8) Code Conversion 6, 7 or 8 bit

(A) Look-Up LT N XXXXX...

LT = Do table look-up code conversion on transmitted data.

LR N XXXXX...

LR = Do table look-up code conversion on received data.

N = Number of characters in look-up table. (N = 256(8-bit), 128(7-bit), 64(6-bit).

XXXXX... = look-up table (1 character/byte).

(B) Shifted Look-Up

ST U D N XXXXX...

ST = Do shifted look-up code conversion of transmitted data, (Implies 7 or 8 bit in, shift character out)

SR U D N_UYYYY...N_DYYYY..

SR = Do shifted look-up code conversion of received data, (Implies shift character in, 7 or 8 bit out)

U = Upshift character (binary)

D = Downshift character (binary)

N = Number of characters in 7 or 8 bit table. (High order two bits indicate upshift/downshift character and both)

N_U = Number of characters in upshifted table.

N_D = Number of character in downshift table.

YYYY = Characters shift table.

Algorithm for Sending and Receiving Information

Unique 22X7 device addresses are provided for transmitting data lines from the 2200 to 22X7 and receiving accumulated data from the 22X7. A fourth unique address is provided for scanning (receiving a fixed block of information with complete status of the 22X7).

The 22X7 normally operates in either an input or an output mode of operation at a given time. (except for receiving break and echo-plex during output).

Use of the output address which is used to transmit a message line from the 2200 to the 22X7 implicitly clears any input operations and buffered input data and initiates output. (including 202 turn-around). Input mode is generally initialized by sending a request to the 22X7 via a "dynamic parameter" via the parameter loading address. It is also possible via "dynamic parameters" to cause the 22X7 to "turn-around" automatically to input after the last current buffer output character is transmitted. The following dynamic parameters are available for input/output mode selection.

B. Dynamic Parameters (Address XXXXXX00)

(1) Transmit a Break - TB

Cause the 22X7 to transmit a 200 ms break as per the current modem selection. (200 ms mark 103, 202, or 200 ms mark on reverse channel, 202 with reverse channel).

(2) Input Selection -

$$\underline{I} \begin{Bmatrix} L \\ A \\ 1 \end{Bmatrix} [T] [E] \text{ echo-plex? } [TO] - 30 \text{ sec. } \text{time out}$$

L -- Select 22X7 for input operations and/or when enabled with input address transmit next complete input line.

A -- Select 22X7 for input operations and/or when enabled by input address transmit all current characters received as per last 2200 scan.

1 -- Select 22X7 for input and/or when enabled by input address transmits one input character or stay not ready.

T -- Following transmission of last input line to the 2200, turn line around for output (202 convention)

E - echo-plex

(3) Output Selection - 0

$\left\{ \begin{array}{l} N \\ T \end{array} \right\} [E]$

echoplex?

N -- Select 22X7 for Output operations.

T -- Select 22X7 for Output operations and after next line received for output and transmitted, turn-around for input operations. (Previous input selection).

(4) Reverse Channel Set RC [i] Set reverse channel output level to zero or one. (i = 0 or 1)

3. Status Scan (Address XXXXXX01)

At all times during the operation of the 22X7, the 2200 can select this address to receive a fixed block of status information. Scans are typically done by the 2200 on a regular basis after input or output operations has initiated, to determine completion and/or errors. This information will typically serve the following functions:

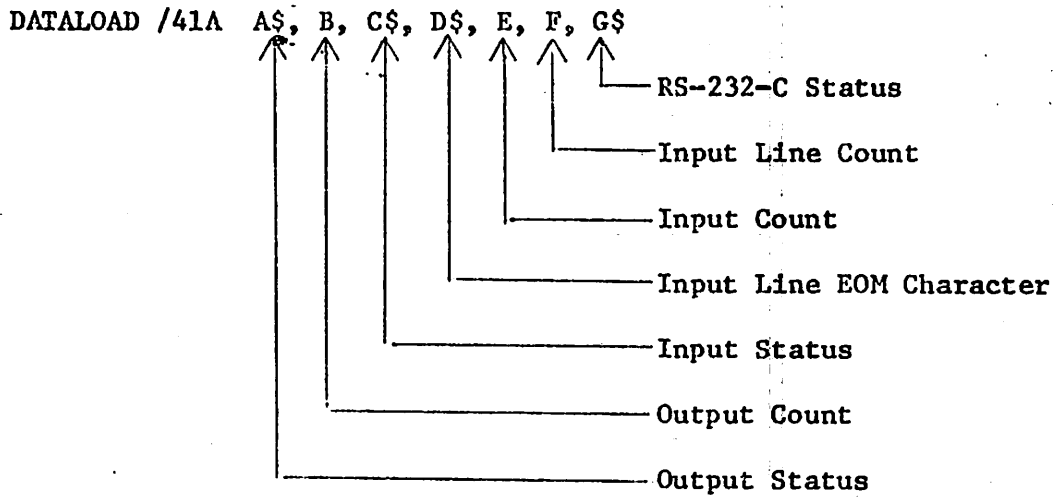
- (1) For Output - To indicate the current count of the number of characters yet to be transmitted from the buffer.
- (2) For Output or Input - To indicate if a break has been received.
- (3) For Input - To indicate the total of characters currently received in the buffer, whether or not a full line has been received (as per specified EOM character), and if so the number of characters in that line. Also sent is error information associated with that line parity, framing, overrun and EOM characters received.
- (4) General Information Associated with the RS-232-C connector and 22X7 status.

RS-232-C - Bits indicating current level of:

- (a) CLEAR TO SEND
- (b) DATA SET READY
- (c) CARRIER DETECTOR
- (d) REVERSE CHANNEL RECEIVING LEVEL
- (e) REVERSE CHANNEL TRANSMISSION LEVEL SETTING

22X7 Status - Mode - Input/Output

The format of the data block sent by the 22X7 with the SCAN statement should be arranged for efficient separation and processing of the data. One suggestion would be to format and separate the information sent such that it can be read by a multi-argument teletype DATALOAD statement for example a standard SCAN statement could be:



where the information sent by the 22X7 would be

- Output Status — One binary character with bits indicating the following:
 - Characters still in 22X7 to be transmitted
 - A Break has been received
 - Carrier Drop or Data Set not ready or no Clear to Send

In all cases 1 = YES, 0 = NO

- Output Count — Three ASCII numeric characters indicating remaining number of characters to be sent.
- Input Status — Two binary characters with bits indicating the following:

1st character

- Characters Received
- Break Received
- Carrier Drop or Data Set Not Ready
- Full Line Received
- Parity Error in Received line or characters
- Framing Error in Received line or characters
- Overrun Error in Received line or characters

In all cases 1 = YES, 0 = NO

*Bit for 30
See
time-out*

- . Input Line EOM Character

EOM character received with full line

- . Input Count -- Three ASCII characters indicating total number of characters received.
- . Input Line Count -- Three ASCII characters indicating number of character in next full line.
- . RS-232-C Status -- One binary character with bits indicating the level of:
 - . Clear to Send
 - . DATA Sent Ready
 - . Carrier Detector
 - . Reverse Channel Receiving Level
 - . Current Setting of Reverse Channel to transmit level.

4. Message Output (Address XXXXXX10)

This will generally consist of a DATASAVE BT or PRINT statement to send a message to the 22X7 for transmission with termination implied by the disabling of the board. Note, that with the DATASAVE BT, trailing spaces can be stripped by the 22X7 microprocessor via parameter TT.

Example:

```
10 DIM A$(2)64
...
...
100 DATASAVE BT /41B, A$()
```

5. Message Input (Address XXXXXX11)

This device address is used for message Input from the 22X7 buffer to the 2200. When enabled, the 22X7 will send to the 2200, a complete message line, all characters currently in the input buffer on the last scan, or one character, depending upon the input mode currently selected in the 22X7. The EOM character is or is not sent, (it is available in the scan input). Error/status/break information pertains to the line or characters received also sent on the prior scan.

A typical 2200 input sequence consists of the SCAN statement to determine if a line has been received and the message input statement in. The teletype or paper tape DATALOAD BT statement will be typically used for receiving the line although KEYIN and \$GIO will also work.

Example:

```
100 DATALOAD /41A A$, B, C$, D$, E, F, G$      (Scan)
110 IF C$ = HEX(09) THEN 200  (Branch if full input line
                               received)

200 DATALOAD BT (N = F) /41D, H$() Receive Line
                                   (Number of characters
                                   received in variable F
                                   in scan)
```

VI. SELF-CHECKING

A mechanism for self-checking of the 22X7 should be provided, i.e., looping the output signal into the receiving portion. One of two possibilities could be used.

- 1) Provide each serviceman with a simple connector plug which routes the output of the input leads. (This is done in manufacture testing of the 2227). This means does not add to the complexity of the board. Operation with echo-plex plus RS-232-C signal checks could then be made.
- 2) Provide a switch on the board to do the above.

VII. TIMING AND PERFORMING CONSIDERATIONS WITH BUFFERING

Special buffering of incoming data characters in a controller board would allow saving of data 'on the fly' to a variety of 2200 peripherals including cassette, disk, or printer.

1. Speed of Wang 2200 peripherals including 150 msec overhead pad.
 - A) 2217 Cassette -- 950 milliseconds to Write a 256 byte cassette record.
 - B) 223X Disk -- 158 milliseconds to Write a 256 byte sector.
 - C) 224X Disk -- 171 milliseconds to Write a 256 byte sector.
 - D) 2261 Printer at 125 lines per minute requires .4 seconds per line.
 - E) 2231 Printer requires 112 to 680 milliseconds per line.

2. Speed of Data Block Reception (Milliseconds)

BAUD	128 BYTES	256 BYTES
300	3840	7680
1200	1152	2304
1800	768	1536
2400	576	1152

The above data implies the following capability with Wang peripherals.

<u>PERIPHERAL/OPERATION</u>	<u>300</u>	<u>1200</u>	<u>1800</u>	<u>2400</u>
1. Record incoming data on cassette, Diablo or Floppy disk	YES	YES	YES	YES
2. LIST incoming data on 2261	YES	YES	YES*	MAYBE*
3. LIST incoming data on 2231	YES	YES*	MAYBE*	NO
*RESTRICTIONS CONCERNING A NUMBER OF SHORT LINES				
4. Listing on 2201/2202 is	ACCEPTABLE AT 110 BAUD, MARGINAL AT 134.5			

WANG

LABORATORIES, INC.

MEMORANDUM

TO: Dr. Wang, Bob Kolk, Norman Lourie, Ed Lesnick, Harold Koplow,
Harry Rothmann, Tyler Olson, Les Haley, S. K. Ho, Henry NG

FROM: Fritz Eberle

DATE: January 28, 1975

SUBJECT: 2200 Asynchronous Telecommunications Controller

INTRODUCTION

This memo describes the hardware controller being developed for synchronous telecommunications as it would look if asynchronous capability were added to it. The additions to the controller hardware to enable asynchronous operation do not appear to be difficult. The described controller would give the 2200 very powerful communications capability. The 2200 interface in the described controller uses one device address and requires a 2200-B with option 2.

There is one major distinction between the Bisync communications system being developed and the proposed requirements for a buffered asynchronous telecommunications controller.

The Bisync system being developed is not a 2200 peripheral in the ordinary sense. It is a system consisting of three parts designed as a unit for use only with each other:

- (1) The programmable controller
- (2) The controller microprogram
- (3) The 2200 operating program

The 2200 is more or less a fourth component of the system. The Bisync package is not designed to be modified by customers. It will contain several versions which can be generated by a 2200 utility program supplied as part of the system. The system initially will emulate the operation of an IBM 2780 terminal and will eventually emulate the 3780 and possibly other Bisync terminals.

Because of the diversity of terminal types and line protocols using asynchronous communications, the proposed approach to asynchronous telecommunication software support is different. It calls for a fixed parameter-driven controller microprogram to be supplied with the controller. The controller microprogram would enable the controller to transmit and receive single lines of data leaving the line protocol logic to the 2200. The 2200 resident communications programs would use the communications controller like any other peripheral device and could be developed by the Wang applications programming group or by customers.

II. 2228 SYNCHRONOUS/ASYNCHRONOUS COMMUNICATIONS CONTROLLER

Controller Microprocessor

The heart of the 2228 synchronous/asynchronous communications controller is an INTEL 8080 microprocessor which is interfaced to the other elements of the controller as shown in figure 2. Input and output instructions in the 8080 contain an 8 bit immediate operand of which the high order 5 bits perform individual control functions and the low order 3 bits specify the input/output function as follows:

bit 7 Clear Reset-Key-Operated

A value of one on immediate operand bit-7 of any input or output instruction causes the Reset-Key-Operated 2200 status bit to be set to zero.

bit 6 Start Receiver Sync Search

A value of one on immediate operand bit-6 of any input or output instruction causes the receiver to abandon synchronization and begin searching the received bit stream for a match with the sync character. Synchronization is re-established when a match is found.

bit 5 Clear Clock-Tick

A value of one on immediate operand bit-5 of any input or output instruction causes the Clock-Tick bit to be set to zero.

bit 4 ENDI 2200 output bit

The value of immediate operand (bit-4) of the output instruction Output 2200 Data sets the ENDI bit of the 2200 input bus. On any other input or output instruction bit-4 is ignored.

bit 3 Clear Underflow/Overflow/Parity-Error/Framing-Error

A value of one on immediate operand bit-3 of any input or output instruction causes the Underflow, Overflow, Parity-Error, Framing-Error, data-link status bits to be set to zero.

bits 2/0 Input/Output Operation Code

The low order three bits of the immediate operand of an input or output instruction define the input or output operation to be done.

Input/Output Control

The Input/Output Operation Code defines the function of output instructions as follows:

000 Output Data-Link Control

The following data-Link control bits are set to the values specified by 8080 accumulator bits:

- bit-0 Request-to-Send modem signal
- bit-1 Data Terminal Ready modem signal
- bit-2 Transmitter Interrupt Enable
- bit-3 Receiver Interrupt Enable
- bit-4 Supervisory Transmit Data modem signal
- bits-5/7 Clock Rate Selection (values in baud)

	/1	/16	/32	/64
000	1758.8	109.9	55	27.5
001	2153.3	134.5	67.25	33.62
010	2400	150	75	37.5
011	3200	200	100	50
100	19.2K	1200	600	300
101	115.2K	7200	3600	1800
110	153.6K	9600	4800	2400
111	x	x/16	x/32	x/64

(x is the clock signal supplied by the modem signals Transmitter Signal Element Timing and Receiver Signal Element Timing)

001 Output 2200 Data

The contents of the 8080 accumulator is output to the 2200 input bus with an IBS strobe. The value of immediate operand bit-4 of the output instruction sets the ENDI bit of the 2200 input bus.

010 Output Transmitter Fill Character

The contents of the 8080 accumulator is output to the transmitter fill character register.

011 Output Transmitter Data

The contents of the 8080 accumulator is output the the transmitter input buffer. The Transmitter Ready status bit is set to zero.

100 Output Display Character

The contents of the 8080 accumulator are output to the display connector with a Data Output strobe. The accumulator data has the following meaning:

bits - 0/6 ASCII graphic character code
bit - 7 Display Home strobe

101 Output Receiver Sync-Match Character

The contents of the 8080 accumulator is output to the receiver sync-match character register.

110 Output Transmitter Control

The following transmitter control register bits are loaded from the 8080 accumulator:

bits 0/1 Mode Selection

00 asynchronous with 1 stop bit
01 asynchronous with 2 stop bits (1.5 @ 5 bit characters)
10 synchronous
11 isochronous with 1 stop bit

bits 2/3 Character Size

00 5 data bits
01 6 data bits
10 7 data bits
11 8 data bits

bits 4/5 Parity Control

00 odd parity
01 even parity
10 no parity
11 no parity

bits 6/7 Clock rate divider

00 nominal rate
01 nominal rate divided by 16
10 nominal rate divided by 32
11 nominal rate divided by 64

111 Output Receiver Control

The following receiver control register bits are loaded from the 8080 accumulator:

bit 0 Mode Selection

0 asynchronous mode
1 synchronous mode

bit 1 not used

bits 2/7 Same as transmitter control register bits

The input/output operation code defines the function of input instructions as follows:

000 Input Data-Link Status

The following data-link status bits are input into the 8080 accumulator:

bit-0 Underflow indication from transmitter
bit-1 Transmitter Ready status
bit-2 Receiver Ready status
bit-3 Overflow indication from receiver
bit-4 Clear-to-Send modem signal
bit-5 Received Line Signal Detector modem signal
bit-6 Data Set Ready modem signal
bit-7 Clock-Tick status

001 Input 2200 Status

The following 2200 status bits are input into the 8080 accumulator:

bit-0 OBS-Strobe-Received status
bit-1 CBS-Strobe-Received status
bit-2 Reset-Key-Operated status
bit-3 2200 Ready signal

010 Input 2200 Data

The contents of the 2200 data register is input into the 8080 accumulator. The OBS-Strobe-Received and CBS-Strobe-Received status bits are set to zero and the Device Ready signal to the 2200 is set to one.

011 Input Receiver Data

The contents of the receiver output buffer is input into the 8080 accumulator. The Receiver Ready status bit is set to zero.

100 Input Asynchronous Mode Status

The following data-link status bits relating only to asynchronous operation are input into the 8080 accumulator:

- bit-0 Parity error
- bit-1 Framing error
- bit-2 Received Data modem signal
- bit-3 Supervisory Received Data modem signal

Interrupt Control Logic

The controller contains logic which allows the 8080 microprocessor to be interrupted. When the 8080 Interrupt signal has a value of one and the controller program has enabled interrupts a subroutine trap to address 0010 will occur. If Receiver Interrupt Enable and Receiver Ready both have a value of one or if Transmitter Interrupt Enable equals one and Transmitter Ready equals one then Transmitter Interrupt Enable and Transmitter Ready both have a value of one, the Interrupt signal will have a value of one. Otherwise the Interrupt signal will have a value of zero.

8080 Reset-Control-Logic

The controller contains the logic to reset the 8080 microprocessor. When the 8080 Reset line is strobed the controller program traps to address 0000. An 8080 reset occurs when a CBS strobe is received from the 2200 with the high order bit of the 2200 output data equal to one. Any controller hardware requiring a master reset is also reset at this time.

Controller Memory

The controller memory consists of 256 bytes of PROM which starts at address zero and 4K bytes of RAM which starts at address 8K. Memory operates with a cycle time of 1.5 microseconds.

Clock

The controller contains a real-time clock which sets the Clock-Tick data-link status bit every 8 milliseconds. The Clock-Tick bit remains set until cleared by the Clear Clock-Tick control bit of an input/output instruction.

Display-Connector

The controller contains a connector (IC socket) for interfacing to a character serial external (CRT) display output device. Data is output directly from the 8080 to the display connector. The logic for driving the display is contained in the Bi-sync tester.

2200 Interface

The 2200 communicates with the controller by enabling the controller and executing "commands" which are the logical unit of conversation. A command consists of a series of back and forth byte transfers between the 2200 and the controller. A byte transfer is described by the notation:

device(strobe-type, data-type)

The four byte transfers operate as follows:

2200(CBS, data-type)

The 2200 outputs 8 data bits into the controller's 2200 data register with an CBS strobe. The CBS-Strobe-Received status bit is set to one and the Device Ready signal to the 2200 is set to zero. If the high order data bit is a one, the 8080 Reset signal is strobed causing a trap to address 0000 in the controller program. The 2200 should wait for Device Ready to be equal to one before outputting.

2200(OBS, data-type)

The 2200 outputs 8 data bits into the controller's 2200 data register with an OBS strobe. The OBS-Strobe-Received status bit is set to one and the Device Ready signal to the 2200 is set to zero. The 2200 should wait for Device Ready to be equal to one before outputting.

CONT(IBS, data-type)

The 8080 outputs 8 data bits to the 2200 input bus with an IBS strobe and the ENDI bit set to zero. The 8080 should wait for 2200 Ready to be equal to one before outputting.

CONT(ENDI+IBS, data-type)

The 8080 outputs 8 data bits to the 2200 input bus with an IBS strobe and the ENDI bit set to one. The 8080 should wait for 2200 Ready to be equal to one before outputting.

When the 8080 inputs the contents of the 2200 data register the status bits CBS-Strobe-Received and OBS-Strobe-Received are set to zero and the Device Ready signal to the 2200 is set to one.

When the 2200 operates the RESET key the 2200 status bit Reset-Key-Operated is set to one. It remains set until cleared by an input or output instruction having the control bit Clear Reset-Key-Operated set to one.

The current value of the status bit 2200 Ready is input by the 8080 when it inputs 2200 status bits. If the 2200 does not currently have the controller enabled 2200 Ready will be equal to zero.

Synchronous/Asynchronous Receiver

The synchronous/asynchronous receiver contains a character output buffer register, a sync-match character register and a serial input receiver register. The sync-match register is loaded by the 8080 and the output buffer is read by the 8080. In order to receive data in synchronous operation, the receiver must establish character synchronization with the incoming bit stream in the following way: (8 bit word length with parity inhibited is assumed)

- (1) The receiver abandons synchronization and starts searching the incoming bit stream for a sync character when the 8080 executes an input or output instruction with the Start Receiver Sync Search control bit set to one.
- (2) Each time the modem signal Receiver Signal Element Timing changes value from one to zero the value of the modem signal Received Data is shifted into the receiver register as a received bit.
- (3) When a bit is received, the receiver register is compared with the sync-match character register. When they are equal, the receiver synchronizes to the bit stream and will read the next 8 bits received as the first received character.

Once synchronization is established, contiguous characters are received in the following way:

- (1) Bits are read into the receiver register as in (2) above. When 8 bits have been read, the contents of the receiver register are transferred to the output buffer and the Receiver Ready status bit is set to one.
- (2) Normally the 8080 will have read the contents of the receiver output buffer before the receiver register becomes full setting Receiver Ready status to zero. If the 8080 fails to read the receiver output buffer by the time the receiver register becomes full, the contents of the receiver register is transferred to the receiver output buffer and the Overflow status is set to one. The Overflow status bit must be cleared by execution of an input or output instruction having the control bit Clear Underflow/Overflow set to one.

Synchronous/Asynchronous Transmitter

The synchronous/asynchronous transmitter contains a character input buffer register, a fill character register and a serial output transmitter register. The input buffer and fill register are loaded by the 8080. During normal synchronous mode operation, contiguous characters are transmitted in the following way: (8 bit word length with parity inhibited is assumed)

- (1) The contents of the input buffer are transferred to the transmitter register and the Transmitter Ready status bit is set to one.
- (2) As long as the modem signal Clear-to-Send has a value of one, the contents of the transmitter register modulates the modem signal Transmitted Data as a NRZ waveform clocked by the zero to one transition of the modem signal Transmitter Signal Element Timing. If Clear-to-Send has a value of zero, Transmitted Data is held at a value of one.
- (3) Normally the 8080 will have loaded the input buffer before the transmitter register becomes empty. When the input buffer is loaded the Transmitter Ready status bit is set to zero. If the 8080 fails to load input buffer by the time the transmitter register becomes empty, the contents of the fill register are transferred to the transmitter register and the Underflow status is set to one. The Underflow status bit must be cleared by execution of an input or output instruction having the control bit Clear Underflow/Overflow set to one.

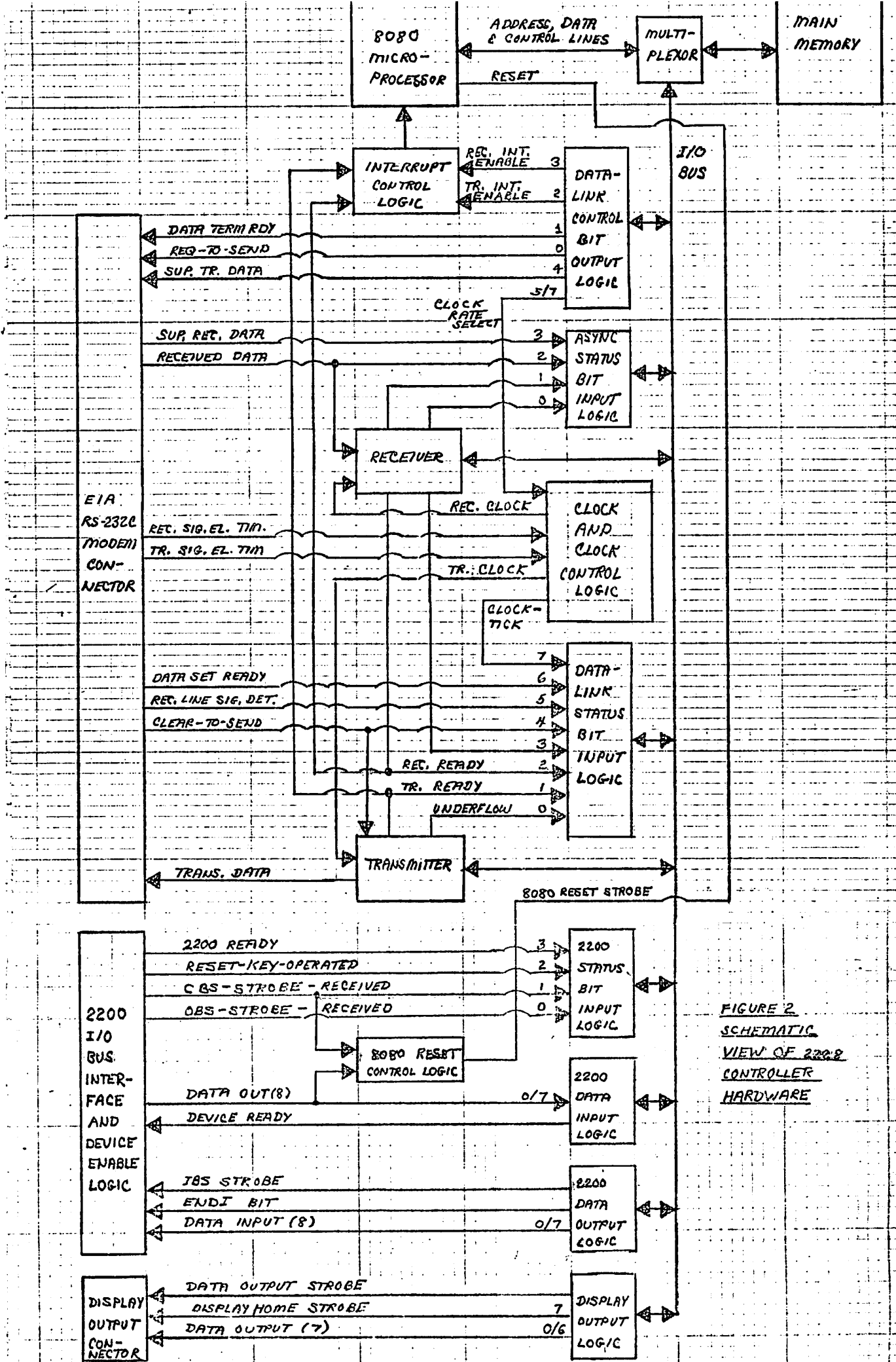


FIGURE 2
SCHEMATIC
VIEW OF 2202
CONTROLLER
HARDWARE

Bob Kolk

WANG

LABORATORIES, INC.

TO: DR. WANG
FROM: HAROLD KOPLow
SUBJECT: COMMENTS ON BOB KOLK'S 22X7 BUFFERED ASYNCHRONOUS
TC MEMO OF 1/14/75
DATE: JANUARY 30, 1975

This memo is divided into two parts. Part I is a critique of Bob Kolk's memo; Part II is the results of a meeting held on Friday, January 24 at 2:00 p.m. in Dr. Wang's office.

PART I

Bob Kolk's document was very well written and seems to more than cover all of Ed Lesnick's requirements as stated in a previous memo. However, there are a number of comments that I would like to make which can be subdivided into two major categories - User Specifications - Hardware Specifications.

I. USER SPECIFICATIONS

Most users of our equipment (both salesmen and customers) are rather ignorant in the various aspects of telecommunications. Therefore, it would be desirable to couple our hardware with program products to facilitate both sales and service. These packages could be narrowly defined.

- a. Teletype Emulator - This package would query the user and tailor the program accordingly. (Note: A teletype is a full duplex product with no special end-of-message characters). A one-to-one correspondence between a teletype and the various 2200 peripherals could be made, i.e. a punch could be a disk, a reader could be a tape, the keyboard could be the keyboard of the 2200, and the printer could be the CRT or the 2201, etc.

The program should be built in such a fashion that the special function keys would turn the punch analog on and off, send an interrupt, prepare data offline for later on-line transmission, etc. Also, there should be a utility which would allow data stored in other formats in the disk to be reformatted so that it could interface properly with the teletype emulator - be made to look like punched data for the teletype emulator.

- b. 2741 Emulator

WANG

LABORATORIES, INC.

TO: Dr. Wang
PAGE: 2
DATE: January 30, 1975

- c. 2740 Emulator
- d. TC 500 Emulator
- e. 2200 to 1200 Emulator - This emulator would work in the same fashion as a 1200 transmitting to another 1200 at high speed. In this case, one of the 1200's would be replaced by a 2200.
- f. 2200 to 2200TC Emulator. Since the 2200 is capable of sending to another 2200, it would be advantageous to employ one line discipline (the same method for blocking of records, end-of-message characters, etc.).

In addition to the emulator packages, a set of TC utilities should also be programmed. This would enable the more sophisticated user to build a TC program without getting down to the DATALOAD BT level. Also, a user would not necessarily have to wade through a TC Manual, which out of necessity, would be as complex as the GIO Manual.

The program products, together with a well written TC Manual for asynchronous communication, should allow us to penetrate this market more successfully.

II. HARDWARE SPECIFICATIONS

I propose that the 22X7 Controller be as similar as possible to the Bi-synchronous Controller. Telecommunication is a rather strange beast - and unless there exists some mechanism to modify the programs of the 22X7, we will be continually remasking or reprogramming chips. The additional cost of employing RAM as in the bi-synchronous controller for the 22X7 program, should be carefully weighed before it is scrapped. Also, much work has already been expended on the hardware and software design for the bi-synchronous controller - I believe much of that design could be utilized on the asynchronous controller. It would also be most desirable to standardize on one LSI Processor.

On the last page, is a chart of operation that can be performed at various baud rates. At what baud rate is it possible to both print and record incoming data on the various devices? The whole purpose of Ed's memo was to specifically point out that this could not be done with the present controller.

WANG

LABORATORIES, INC.

TO: Dr. Wang
PAGE: 3
DATE: January 30, 1975


The saleability of a TWX-Telex emulator, which could be coupled with a program product, is questionable. Will the effort required to support this be worth the cost?

Bob has made a very good and explicit specification (which could if required be made less sophisticated) for an improved controller. However, I must reiterate that it should be packaged with program products or we will be continually asked to support or to provide customized programs for customers in order to be able to communicate with the various devices now on the market; or as in the case of GIO, few questions will be asked because it would be impossible for the average user to employ this new controller.

PART II

On Friday, a meeting was held in Dr. Wang's office to specifically discuss this new 2200 buffered asynchronous telecommunication interface. The following was resolved:

- a. The 22X7 buffered hardware device will use the same microprogrammable hardware as the by-synchronous controller.
- b. Fritz Eberle with Tyler Olson and Les Haley will manage the microprogramming of the hardware product.
- c. Harry Rothmann - Bill Lynes will manage the program product.
- d. The offering will be a combined hardware/program product.



Harold Koplow *db*

HK:dg:1975-82

cc: Ted Babine
Fritz Eberle
Bob Kolk
Ed Lesnick
Norman Lourie
Bill Lynes
Tyler Olson
Harry Rothmann

MEMO TO: Bill Lynes
FROM: Joe Handy
SUBJECT: 2227 BOARD MODIFICATION FOR 'BREAK' USE
DATE: October 13, 1975

*To Fred Wang
for review!*

Bisync

Pombean

1. On transmitting a break, it is suggested that the 2227 board be modified to allow variable hex digits to trigger the break. The hex digit should be able to be set physically by the operator similar to the RCV or KMT method of setting. Once set, transmitting that particular hexdigit should activate the break. Also, it is suggested that a hexsetting of 00 signify "no break desired" and consequently the transmitting of all hexdigits would result without break activation.

2. On receiving a break, it is suggested that no modifications be made to the 2227 board. That is because of the following assumptions:

- a) The 2200 would operate in a half-duplex system environment.
- b) The 2200 would receive a break signal only when the 2200 was transmitting data, not while it is receiving data.
- c) While the 2200 was transmitting data, the most common data received would be an acknowledgement, non-acknowledgement, or break.
- d) A par/fram error received during 2200 transmission would indicate a break almost 200% of the time since practically nothing else is being received, and par/fram errors are not that common.

Joe

JH/dt

Brayne

WANG LABORATORIES, INC.

ROBERT S. KOLK

MEMORANDUM

TO: Dr. Wang
cc: Norman Lourie, Fritz Eberle, S. K. Ho, Henry Ng, Carl Masi,
Al McDonald, Ed Lesnick, Harry Rothmann, Bob Rainville,
Joe Handy, Tyler Olsen, Bruce Patterson
FROM: Bob Kolk
DATE: October 24, 1975
SUBJECT: Specifications 2227B (Buffered Asynchronous Controller)

Enclosed is a specification for an improved version of the 2227 asynchronous T. C. controller. The logic for it could be used for either a plug-in controller board for the 2200 S/T or as built-in logic or internal plug-in logic on the integrated version of the 2200 (2200P).

It is worthwhile to briefly discuss how this relates to the new 2228 synchronous controller.

1. The 2228 has an 8080 microprocessor and 4K of RAM, both of which are necessary to support complex synchronous protocols. The selling price of this board will be over \$2000, an acceptable price for speed synchronous T. C. capability.
2. Many competitors however, offer asynchronous capability at prices well under \$1000. If we were to offer a version of the 2228 for just asynchronous, we would be forced to offer it at a much lower price and have less margin. In addition, problems associated with asynchronous are not always easily solved by a dedicated microprocessor, since they often relate to special programming related to termination characters, padding delays, sign-on conventions unique to particular asynchronous terminals and/or time-sharing networks. These types of problems are often more easily handled with an easy to program language such as BASIC as opposed to 8080 microcode.
3. The major problems associated with the 2227 which made the programming so complex was essentially the lack of multi-character buffering. Without multi-character buffering, the controlling BASIC program designed is relatively complex because it must be required to receive an incoming data block on a character-by-character basis and in between characters, attempt to process, print, or save the data. In addition, incoming data stream which is continuous cannot be saved on tape or disk since, that required a discrete amount of time away from the receiver logic to write records on tape or disk.

4. Although this memorandum deals with the possibility of a newly designed controller board, the use of a 2228 should not be entirely removed from consideration. It would be possible to rapidly develop an asynchronous version of the board, (possibly only a change to the bootstrap PROM) and having similar boards for manufacturing and service is an advantage. It would also provide substantially greater buffering capacity which could be an overall advantage. (Higher continuous stream baud rates could be supported). If the 2228 were used, the approach might be to implement all microcode on the PROM bootstrap chip without microcode loading.

If indeed, there is not sufficient cost saving in a design eliminating the 8080 microprocessor, the 2228 design might be utilized, although it is possible that a buffered 2227 card could approach half the cost of the 2228.

SPECIFICATIONS

I. INTRODUCTION

Although the general requirements to support an asynchronous design could be much less complex than bisynchronous, the following capabilities are probably important.

1. The ability to set baud rate, parity (even, odd or none), number of data bits, number of stop bits under program control. This eliminates expensive switches and provides more flexibility.
2. The ability to send and receive breaks.
3. The capability for the controller, BASIC program to sense incoming errors (parity, framing) and certain modem signals. The latter determines proper modem connection, a dropped carrier, etc.
4. The ability to buffer with (FIFO'S) both transmitted and received data. Although the same FIFO could support both, (half duplex), the cost saving may not be worth the possible limitations and complexity. Although true full duplex operation is somewhat less common, additional program/hardware logic is required to determine when switch over from transmission to receiving occurs. That is especially true for 103A type modems where no line turn around is required. For example, when a transmitting FIFO buffer empties, a decision must be made by program/hardware as to whether more information is to be sent, information is to now be received. If both transmitting and receiving are independently buffered, that decision is unnecessary, control is simpler, and more free time could be spent servicing tape or disk. (202C modem operation still requires some decision of this type, although it could be semi-automatic).

The size of the FIFO buffers depend largely upon the design goals. 1200 baud is a reasonable maximum rate to support for asynchronous. The design criteria in effect becomes at what baud rate could a continuous stream of data be received and saved on floppy or cassette. The following chart presents these capabilities:

<u>INPUT</u> <u>FIFO BUFFER SIZE</u>	<u>MAXIMUM BAUD RATE</u> <u>FOR STORAGE ON CASSETTES</u>	<u>MAXIMUM BAUD RATE</u> <u>FOR STORAGE ON FLOPPIES</u>
64	600	2400
128	1200	4800

If it is important to provide the most flexible asynchronous capability with this design, then a 128 character FIFO input buffer is desirable. The output buffer size should probably be matched to the input size, for throughput considerations.

5. The ability to transmit and receive data between the 2200 I/O bus and the FIFO's.
 - Data sent to output FIFO buffer via OBS strobes, when FIFO output buffer ready (Can receive more characters) presented as RBI signals when card enabled.
 - Characters automatically strobed from the FIFO input buffer if characters are in the buffer and if CRB (CPU is ready) and if the card is enabled.
6. Internal design capability to automatically strobe data between the transmitter and receiver character buffers and the input and output FIFO buffers.
7. Device Address -- Generally, a single device address is sufficient since the CRB (CPU ready/busy) level is sufficient to differentiate input from and output to FIFO buffers. However, one bit of the address could be used to identify pending input operations to set or reset REQUEST TO SEND for 202C modem operations.
8. A selectable mode could be made available such that receiving a break character could clear the output buffer, to terminate output operations.

II. HARDWARE DESIGN CONSIDERATIONS

Aside from buffer size, the most important question to be answered is what transmitter/receiver chip is to be used. It would be desirable to use the INTEL chip from the point of view of procurement. However, with the INTEL chip 8-bit data input, 8-bit data output and 8-bit control all share the same 8 pins. Therefore, if the design were strictly hardware without a microprocessor, 3 level multiplexing and possibly some form of scan logic is required. This may be more expensive. An engineering review should evaluate alternatives. The remainder of this specification will deal with the functional aspects of the board, how they are accomplished; these will be largely determined by choice of the transmitter/receiver chip. The specification presented outlines possibly the most efficient hardware approach considering the use of the INTEL 8251. It could vary considerably if other chips were used.

III. FUNCTIONAL SPECIFICATIONS

The 2227B will support the following functions:

(1) The ability to select the following asynchronous transmission characteristics under BASIC program control:

- Baud rate -- 110, 134.5, 150, 300, 600, 1200, 1800, 2400
(Possibly 4800, 7200, 9600 for interface considerations)
- Parity (Yes or No, Even or Odd)
- Number of data bits -- 5, 6, 7, 8
- Number of stop bits -- 1, 1 1/2, 2

If any reset is required for the transmitter/receiver chip would be done in conjunction with sending this control data from the 2200.

(2) The ability to sense the following error/break/modem status/FIFO conditions under BASIC program control.

- Parity Error
- Overrun Error
- Framing Error
- Break Received (125 ms) (Data in line)
- DSR (Data Set Ready)
- CTS (Clear to Send)
- TXE transmitter empty (both hold and shift character buffer empty)
- Receiver Ready (Receiver character buffer has character)
- FIFO output buffer ready (can receive more characters)
- FIFO input buffer ready (has character to send to CPU)
- Possibly reverse channel input level

Sensing error and break status (sent into the 2200) should reset these conditions.

FIFO buffer count could be useful but is not absolutely necessary.

(3) The ability to control the modem output levels and/or reset status conditions.

- Control Data out, supervisory channel out (to send breaks)
- Reset Error and break condition
- Control terminal ready (automatic hangup)
- Control RTS (Request to Send)

The 103A modem ignores RTS. With the 202C operation however, RTS must be high for transmission, and low for receiving of data, with a 200 ms turn around time supplied by the modem. Although we could make RTS level directly controllable by a controlling BASIC, this tends to minimize the time available for overlapped CPU operation and complicated BASIC control programs somewhat. A more desirable solution is to provide two device addresses, (as was done in the 2227). One address is considered an output address and the other the input address. When the board is enabled with the output address a level is set causing RTS to go high if and when the receiver chip (RxRDY reset) and FIFO input buffer have no characters. If the controller is enabled with the input address, a level is set which will cause RTS to go low if and when the transmitter (TxE = high) and FIFO output buffer have no more characters to send. In addition, when a break is detected, the RTS level forcing a low condition is set (in addition to clearing the output buffer).

With this approach maximum CPU overlap can be obtained and program control could be somewhat simplified.

IV. BASIC CONTROL SPECIFICATIONS

It should generally be assumed that the \$GIO statement will be available to support the operation of this controller. This statement has some very good features for supporting asynchronous operations such as the ability to specify multi-terminator, multi-separator characters. The controller hardware design should support the following:

- (1) Device Addresses for Enabling -- 8 address switches for input
8 address switches for output
- (2) Initialization of Controller/Transmitter/Receiver Chip -- The control sequence required to select baud rate, character format, reset errors depends a great deal on the transmitter/receiver chip used. The following scheme is based on the use of the INTEL 8251. Other arrangements could be more desirable for other chips, or these may be a more desirable sequence for the 8251.

CBS -- Control Strobe

There will be one control strobe using CBS which effect sets up a temporary multiplexer condition for 8251 chip to send control commands and/or receiver status. This control byte will generally be followed by another CBS strobe (or a controller produced IBS strobe) which transfers the control/status data and returns.

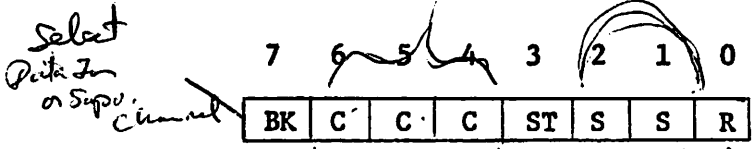
For clarity the control operation will be defined in both an engineering and programming sense.

V. ENGINEERING SPECIFICATION

Possible 8 bits

A. Control

The 8 data bits sent with the CBS strobe have the following meaning:



Break Override
(Inhibit Break Detection)

Information for
Clock Rate Selection

CCC	/16	/64
000	Do not reset CCC or BK to any new value	
001	109.9	27.5
010	134.5	33.62
011	600	150 ✓
100	1200	300 ✓
101	2400	600
110	4800	1200
111	7200	1800

200

7200

Setting for Supervisory
Transmit Channel
0 = 0
1 = 1

1 = Reset 8251 chips (external reset). Also cause FIFO buffers, break to be cleared.

01 = Setup multiplex logic to 8251 chip such that the data presented on the next CBS strobe will be gated to the chips as a mode* instruction or command instruction. Following the completion of that strobe, multiplex and control logic will be reset back to data transfer to the FIFO'S *Mode instruction if R = 1 in control byte

10 = Setup multiplex logic and chip control logic such that a chip status read will be performed and the data strobed to the 2200 by an IBS strobe when the 2200 CRB level (CPU ready/busy) becomes ready. Following completion of the IBS, chip logic will be reset back to data transfer between FIFO'S

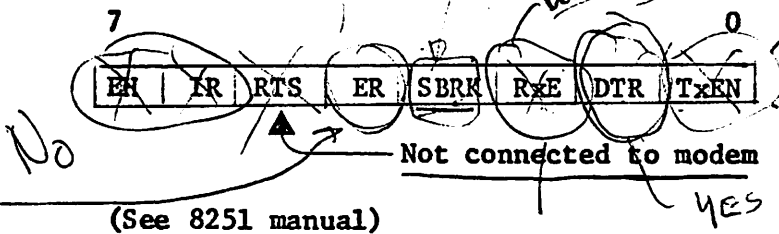
11 = Strobe back FIFO buffer, break information when CRB = ready.

Facility to clear

(1) Command Word

If S, S, R = 0, 1, 0 (the CBS byte word (No Reset)) the next CBS strobe supplies the 8251 command word as follows:

Possibly Disable Receiver



Done Need always enable

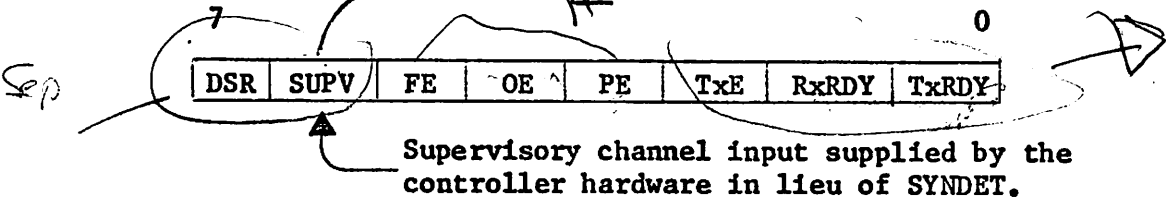
Data terminal Ready Programmable on/off

(See 8251 manual)

(2) Status Read

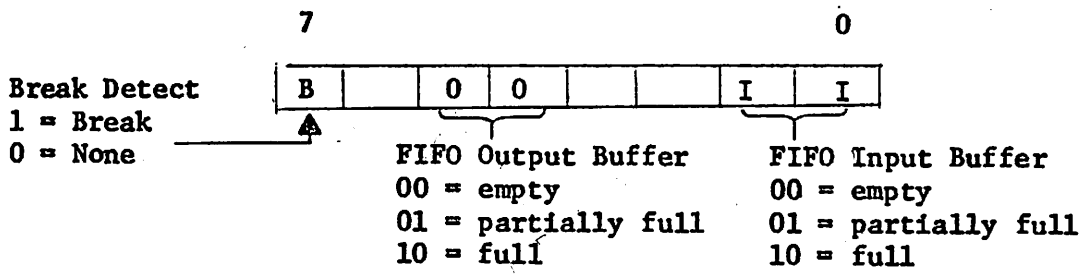
If S, S, M = 1, 0, 0 in the CBS data byte the following 8541 status word will be read and strobed to the 2200 when CRB = 0 (single byte input)

Select Break on Data or Supervisory Channel



(3) FIFO/Break Status

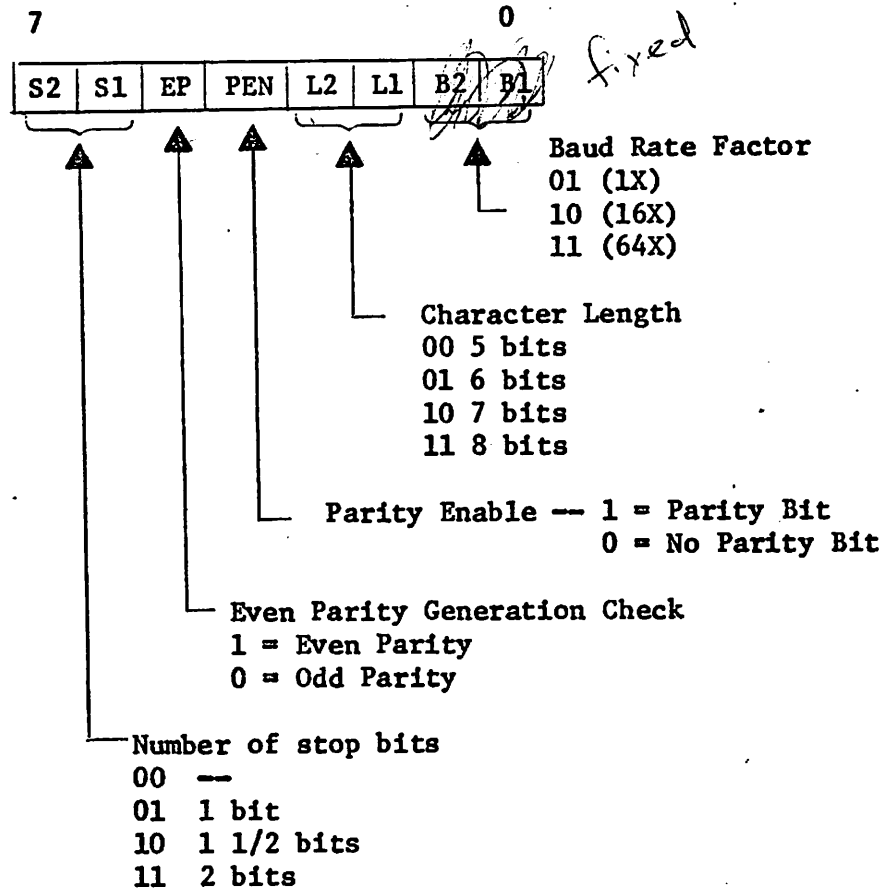
If S, S, M, = 1, 1, 0 in the CBS data byte, the following FIFO buffer status information will be strobed in the 2200 (IBS) when CRB is ready.



Even we have expandable 1K-2K for each

(4) Mode Instruction

If S, S, R = 0, 1, 1 in the CBS data byte word the next CBS strobe supplies the mode instruction to the 8251 as follows:



(5) SBRK

Breaks could be generated by a PG or under program control depending upon how the BK bit is utilized. It is initially proposed that SBRK on 8251 be used since it holds the data out signal on the RS-232 connector to a mark state. This also allows variable length breaks.

(6) Data Transfer

When the controller card is enabled:

- (a) characters will be transferred into the output FIFO buffer via a 2200 OBS strobe when ready (RBI = 1) where ready = FIFO output buffer not full and can accept next data transfer.
- (b) characters will be strobed from the FIFO input buffer into the 2200 by and IBS generated by the controller when the FIFO input buffer has characters to send and CRB (CPU ready/busy = 0 ready).

Whether or not card enabled

- (a) Automatic multiplexer connection and transfer of characters from the receiver character buffer to the FIFO buffer when a character is available and when the FIFO is not full.
- (b) Automatic multiplexer connection and transfer of characters from the FIFO output buffer to transmitter character buffer when characters are available in the FIFO and the transmitter character buffer is empty.

(7) Incoming Break Detection

Incoming break detection will be done by logic independent of the 8251 chip, sensing the RS-232 data in line. If the line is in the mark condition for 125 ms, a break is detected. This will:

- Set a level which can be read via the FIFO/Break Status Input
- Automatically clear the output buffer.
- Force the Request to Send control level to "set low" condition
- The Break is reset when either error reset appears on a command instruction to the 8251 or FIFO/Break Status is read by the 2200.

Break Detection is inhibited if the high order bit of the CBS control byte (BK) is sent with a 1.

(8) Request to Send Control

- When board enabled with output address, set level which sets RTS to 1 when FIFO input buffer empty and receiver not ready (RxRDY = 0).
- When board enabled with input control address or break detected, set level which sets RTS to 0 when FIFO output buffer empty and transmitter empty (TxE = high).

VI. PROGRAMMING CONTROL SPECIFICATION

(1) Initialization

Prior to transmission the following \$GIO statement is executed to initialize the 2227B controller and select desired baud rate and character format.

\$GIO INIT (44-3 44- 4402 44- AS)
 ↑ ↑ ↑ ↑
 (a) (b) (c) (d) (e)

where:

(a) (c) = BAUD RATE/NUMBER OF DATA BITS/BREAK DETECTION

(a)		(c)							
		2	6	A	E	3	7	B	F
9	1	110/5	110/6	110/7	110/8	27.5/5	27.5/6	27.5/7	27.5/8
A	2	134/5	134/6	134/7	134/8	33/5	33/6	33/7	33/8
B	3	600/5	600/6	600/7	600/8	150/5	150/6	150/7	150/8
C	4	1200/5	1200/6	1200/7	1200/8	300/5	300/6	300/7	300/8
D	5	2400/5	2400/6	2400/7	2400/8	600/5	600/6	600/7	600/8
E	6	4800/5	4800/6	4800/7	4800/8	1200/5	1200/6	1200/7	1200/8
F	7	7200/5	7200/6	7200/7	7200/8	1800/5	1800/6	1800/7	1800/8

(b) = PARITY/NUMBER OF STOP BITS

(b)	Parity Bit	Number of Stop Bits

4	No	1 bit
5	Yes, Odd	1 bit

7	Yes, Even	1 bit
8	No	1 1/2 bits
9	Yes, Odd	1 1/2 bits

B	Yes, Even	1 1/2 bits
C	No	2 bits
D	Yes, Odd	2 bits

F	Yes, Even	2 bits

(d)(e) = 37 Enables for transmitting, receiving, resets errors.

(2) Testing Status of Transmitter/Receiver buffers

The status^A of 128 characters transmitter and receiver buffers can be tested at any time after initialization by the following \$GIO statement.

\$GIO TESTB (4406 8601, A\$)

The first character of the alphanumeric variable specified, (A\$) will be set with the hexadecimal value X Y

where:

X = Status of transmitter buffer
0 = empty*
1 = partially full
2 = full
8 = break has been received (transmitter buffer automatically empties and will ignore additional characters sent for output until the break condition is reset).

Y = Status of receiver buffer
0 = empty*
1 = partially full
2 = full

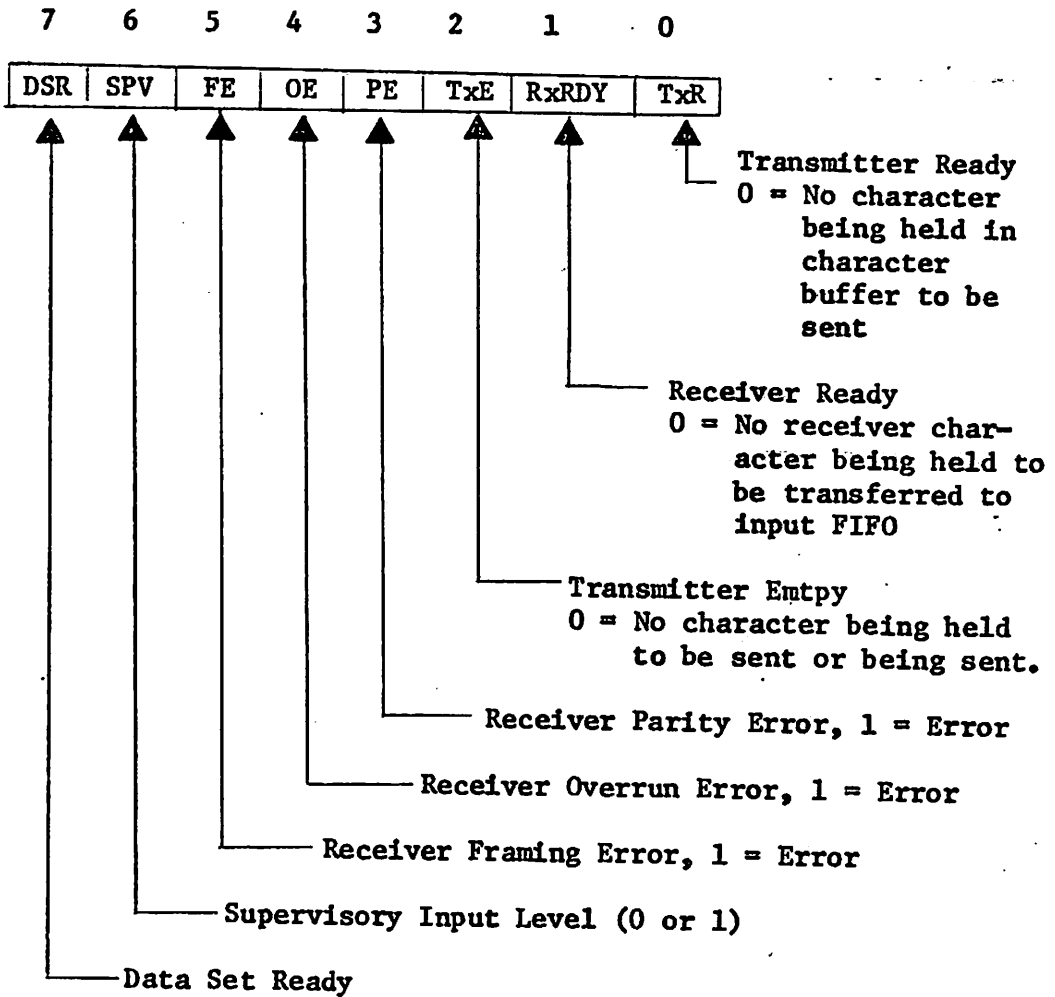
*Design could be such that the buffer empty condition includes both the status of the buffer and transmitter/receiver.

(3) Testing Status of Errors, Transmitter/Receiver Logic, Modem Lines

The following \$GIO command will provide an 8-bit status byte with current information on receiver errors, transmitter/receiver character buffer status. For most asynchronous support programs, the most primary status information which will be useful is data set ready, to insure proper local modem connection and the receiving error bits.

\$GIO STATUS (4404 8601, A\$)

The first character of the specified alphanumeric, (A\$), will be set with the following status information.



(4) Resetting Errors, Break Conditions

The following \$GIO will reset errors and break conditions sensed by the "TEST" or "STATUS" \$GIO statements.

```
$GIO RESET (4402 4437, A$)
```

(5) Transmitting Data

Once the controller has been initialized, (by a \$GIO INIT statement), data can be transmitted by a number of 2200 BASIC statements: PRINT, PRINTUSING, DATASAVE BT, \$GIO. If the 128 characters transmitter output buffer becomes full during the process of transmitting a line, the output statement will be held up until all characters can be sent to the buffer. If it is desired that 128 characters be transferred to the output buffer without instruction delay, the program can be designed to test for a TRANSMITTER BUFFER EMPTY condition before executing an output statement.

If at any time during, transmission, a break is received (125 ms on data in line), and the controller was initialized for Break Detection the transmission buffer will be automatically reset to an empty status and held there until the break condition is reset. Subsequent characters sent by BASIC output statement will be ignored until break is reset. (Reset by a \$GIO TESTB or \$GIO RESET statement). If convensions and breaks are normally anticipated, a \$GIO TEST statement should be executed prior to any output statement.

(6) Receiving Data

Although several BASIC statements can be used for inputting received data from the 128 character receiver buffer (INPUT, KEYIN, \$GIO), \$GIO using the FXXX telecommunications data input microcommand is most desirable because:

- (1) It provides the capability to separate data into different elements of the receiving array by specifying a number of different separator characters.
- (2) It provides for termination of the input sequence by specification of a number of different terminator characters.
- (3) It allows echoing either to CONSOLE OUTPUT or back to the transmitter buffer. (ECHOPLEX transmission).
- (4) It supports a timeout exit from the input sequence.

All of these can be important for various asynchronous protocols.

(7) Transmitting a Break

A break can be transmitted by executing two \$GIO (one to start the break signal and one to end it) with an appropriate delay in between, (corresponding to the desired break length). The following sequence illustrates send a 200 ms break (which is the typical break period).

```
100 $GIO STARTBK (4402 443F, A$) (Start Break)
110 FOR I = 1 TO 40: NEXT I (Delay 200 ms)
120 $GIO STOPBK (4402 4437, A$) (Stop Break)
```

It should be noted that similar control is available for the supervisory channel output with \$GIO (4404, A\$) -- set supervisory output to 1 and \$GIO (4400, A\$) -- set supervisory output to 0.

After a break is transmitted it is usually appropriate to unload the input buffer of any residual character by some input statement, and reset receiver errors (\$GIO RESET).

(8) Device Addresses

- For operations with a 103A type of modem a single device address, /219, is sufficient for all operations.
- For operations with a 202C type of modem the following addresses are generally required.
 - Input -- /219
 - Output -- /21D
- INIT, TESTB, STATUS, RESET
For most efficient operation*, generally the device address of whatever operation will occur next (i.e., /219 for input, /21D for output).

*When going from a transmitting to a receiving operation, if a statement such as TESTB or STATUS is executed with the input address immediately after the last output data is sent to the output buffer, the line is automatically "turned-around" following the transmission of the last output character, and up to 128 characters can be subsequently received with further 2200 intervention.

Slapping at the connect
as per new 2228 ?
and wire and light
as per 2228 ?

MEMORANDUM

Bob Kolk

TO: Telecommunications File
FROM: Fritz Eberle
DATE: February 20, 1976
SUBJECT: Revised 2227B Buffered Asynchronous Telecommunications Controller

This memo is a preliminary specification of how a buffered asynchronous controller based on hardware similar to the 2228 would look to the 2200 programmer. It has the same general capability as the one proposed by Bob Kolk in his memo of October 24, 1975. The use of a microprocessor makes this controller more powerful and easier to program. The specific differences are as follows:

1. The controller rather than being implemented entirely from discrete logic uses a slightly modified version of the 2228 controller. It contains an Intel 8080 microprocessor with fixed microcode residing in 256 or 1K bytes of PROM. Buffers and status variables reside in 1K bytes of RAM. The interface to the 2200 I/O bus and the interface to the modem connector are the same as those on the 2228B.
2. The way in which status, control, and data information is communicated between the 2200 and the controller is different than in the original proposal. The controller has one device address, nominally HEX(1C), rather than two.
3. One marginally useful feature has been dropped - the interface and connector to the Bell 801A automatic calling unit.
4. Several features have been added, being made possible by the use of a microprocessor:
 - a. automatic echoplex operation
 - b. automatic insertion and removal of shift characters
 - c. substitution of characters received in error in error
 - d. detection of receive timeouts
 - e. code translation
 - f. synchronizing the beginning of received data on one of several specified characters.

PRELIMINARY SPECIFICATION
22278

BUFFERED ASYNCHRONOUS TELECOMMUNICATIONS CONTROLLER

The 22278 is a communications controller for serial asynchronous transmission and reception of data at speeds up to 9600 baud. Physically it is a double-card standard 2200 controller which occupies one slot in the chassis of a 2200S, T or WCS processor. It contains a 25-pin connector with an EIA RS-232/CCITT V24 standard modem interface. There are no operating controls or indicators on the controller except for a hole in the middle of the aluminum bar into which an LED can be mounted or a wire connected to indicate the value of the Data-Set-Ready modem signal (a legal requirement in some European countries). A single set of 8 switches on the controller circuit board is used to set the device address which by convention is HEX(1C).

Architecturally, the controller contains an Intel 8080 microprocessor with hard microcode residing in either a 256-byte or 1K byte PROM. 1K bytes of RAM are used for status, code translation tables, and buffer storage. An asynchronous transmitter/receiver chip interfaces the 8080 to the modem transmit Data and Receive Data signals. A selectable-speed clock allows transmission speeds ranging from 110 to 9600 baud (speeds above 1800 baud are useful only for direct wired data links). The 8080 can sense the value of the modem signals Receive Data, Supervisory Receive Data, Data Set Ready and Receive Line Signal Detector. It can set the level of the modem signals Data Terminal Ready, Request to Send, and Supervisory Transmit Data.

Special Features

The controller gives the 2200 programmer several features which in certain situations simplify the transmission and reception of data. Use of any of these features is optional.

Code Translation

Code translation allows the 2200 to interchange data with the controller in the ASCII code set regardless of the transmission code set. The controller contains two 256 byte code translation tables. When these tables have been loaded into the controller from the 2200 and code translation has been enabled by setting the "C" bit to one in the communication control vector, the controller will translate characters before transmission and after reception.

During transmission, a character output by the 2200 is used as an 8-bit index into the 256 entry transmit code translation table. An 8-bit character obtained from the table then placed into the transmit buffer.

During reception, a character received by the controller is used as an 8-bit index into the 256 entry receive code translation table. An 8-bit character is then obtained from the table. If the code translated character is a NUL (X'00') character it is discarded. Otherwise it is placed into the receive buffer. Superfluous characters used for timing or fill can be removed by translating them to NUL codes.

Insertion and Removal of Shift Characters

The controller contains a feature which frees the 2200 program from concern with shift characters when transmitting and receiving data in a code set which utilizes shift characters (for example when emulating an IBM 2741 or communicating with a Wang 1200).

When the "MM" bits in the communication control vector are set to '01' shift characters will be inserted into the transmitted data stream and removed from the received data stream. This feature can be used with transmission data words of 5, 6, or 7 bits and is normally used in conjunction with code translation.

During transmission the controller uses the high-order bit of the (code translated) character in the transmit buffer as a "shift status" bit. One represents an up-shifted character and zero represents a down-shifted character. A shift character is automatically transmitted between any two characters having different "shift status" bits. (An up-shift character is transmitted prior to a character having up-shifted status and a down-shift character prior to a character having down-shifted status.) The "shift-status" bit is not transmitted since the transmission data word consists only of the low-order 5, 6, or 7 bits of the character.

During reception the controller sets the high-order bit of the received character (before code translation) according to the most recently received shift character (one represents up-shift; zero represents down-shift). The up-shift and down-shift character are defined in the transmission code set by bytes 4 and 5 of the communication control vector. If the received character is a shift character it is discarded. Otherwise it is code translated and placed in the receive buffer.

Echoplex Operation

Echoplex operation is a type of error checking used by some computer systems in their support of Teletype terminals. The modems and transmission facilities must be full-duplex. The terminal is expected to immediately retransmit each received character so that the computer system may compare it with the character it transmitted.

Echoplex operation is selected when the "MM" bits in the communication control vector are set to '10'. The controller operates as if in full-duplex mode except that received characters are retransmitted ("echoed"). If a character is received with a parity or framing error the received error substitute character from byte 3 of the communications control vector is transmitted instead of the received character.

Received Data Synchronization

When operating with half-duplex modems and transmission facilities line transients occurring during line turnaround sometimes cause reception of extraneous characters. The 2200 can be freed from the task of detecting and discarding these characters when the first valid data character is known to be one of up to four specific characters. When the "X" bit in the communication control vector is set to '1' receive data synchronization is enabled. Following a line turnaround from transmit to receive, received characters are discarded (and are not echoed if echoplex mode is selected) until a character is received which matches one of the characters in bytes 7-10 of the communications control vector. The first matched character and all subsequently received characters are placed in the receive buffer.

If fewer than four match characters are to be used duplicate characters should be set in bytes 7-10 of the communications control vector.

2200 programs operate the controller by the following group of \$GID statements:

Set Communication Control Vector

\$GID SETCCV/O1C(G0\$,G#)C#()

The communication control vector (CCV) contained in the 10 element array of 1 byte strings C#() is set into the controller. The controller buffers and the communication status byte are cleared.

Read Communication Status Byte

\$GID READCSV/O1C(G1\$,G#)A#

The single byte character string A# is set to the value of the communication status byte. The error and received break indicators in the controller are cleared.

Load Transmit Code Translation Table

\$GID LOADTTBL/O1C(G2\$,G#)C1#()

The transmit code translation table is loaded into the controller from the array C1#(). Execution of code translation by the controller is optional and is controlled by a bit in the CCV.

Load Receive Code Translation Table

\$GID LOADRTBL/O1C(G3\$,G#)C2#()

The receive code translation table is loaded into the controller from the array C2#(). Execution of code translation by the controller is optional and is controlled by a bit in the CCV.

Disconnect

\$GID DISC/O1C(G4\$,G#)G#

The controller disconnects from the line by setting the Data Terminal Ready signal to zero for a period of 3 seconds.

Send Break

\$GID BREAK/O1C(G5\$,G#)G#

The controller sends a 200 millisecond break according to the break control bits of the communication control vector.

Start Receiving Data

+

\$GID STARTRCV/01C(G6\$,G\$)G\$

If set for half-duplex operation the transmit and receive buffers are cleared and the controller enters receive mode and starts receiving data. The receive timeout is started. If the X bit in the communications control vector is set no characters are stored in the receive buffer until a character is received which is equal to one of the receive synchronization characters defined in bytes 7-10 of the CCV.

Input Received Data

+

\$GID RCV/01C(G7\$,G\$)I#()

The contents of the receive buffer are input into the array I#() and byte 10 of G\$ is set to the binary representation of the number of bytes received.

Send Data

+

\$GID SEND/01C(G8\$,G\$)O#(<1,N>

If set for half-duplex operation the receive buffer is cleared. The first N bytes of the array O#() are output to the controller transmit buffer. The controller transmits the data and remains in transmit mode (if set for half-duplex operation).

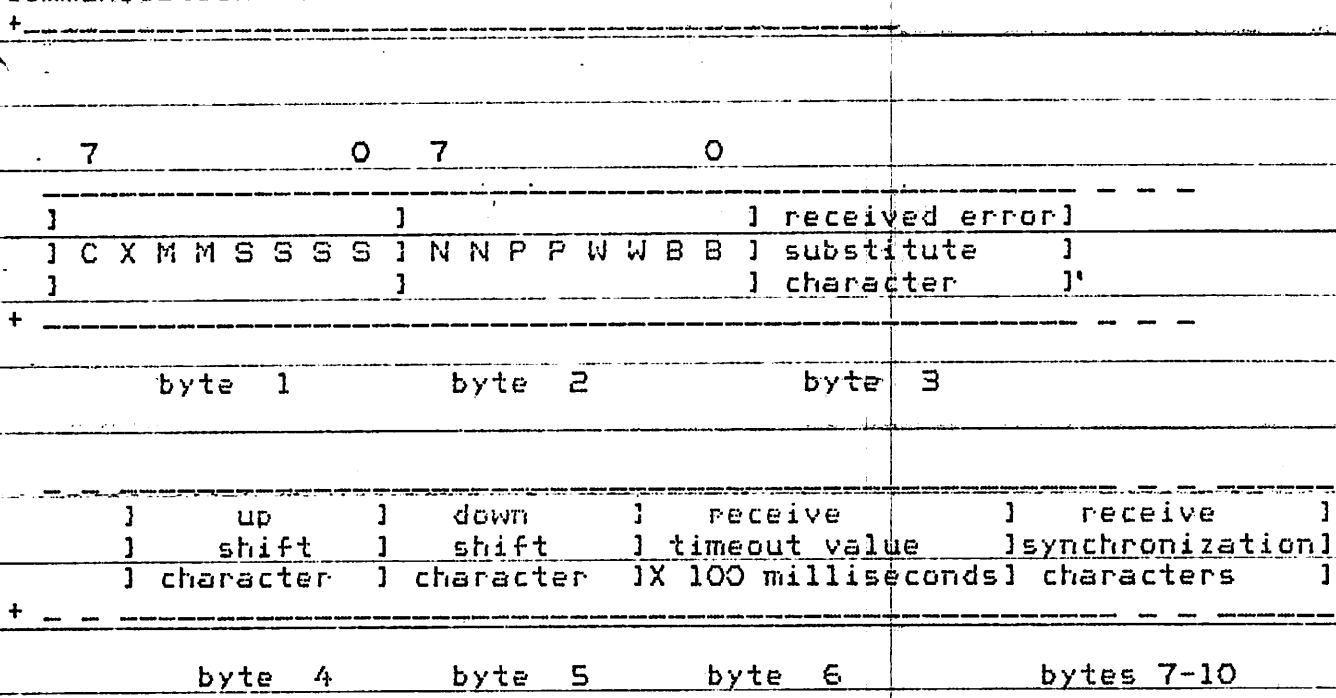
Send Then Receive Data

+

\$GID SENDRCV/01C(G9\$,G\$)O#(<1,N>

This instruction is applicable only for half-duplex operation. The first N bytes of the array O#() are output to the controller transmit buffer. The controller transmits the data and then executes a 'Start Receiving Data' command.

Communication Control Vector Format (10 bytes):



Bit Meanings in Bytes 1 and 2:

- BB = 0 = No break transmission/detection
 1 = Break enabled on transmit/receive data signals
 2 = Break enabled on supervisory transmit/receive data signals
 3 = Inverted polarity break enabled on supervisory transmit/receive data signals
- WW = 0 = 5 bit data word
 1 = 6 bit data word
 2 = 7 bit data word
 3 = 8 bit data word
- PP = 0 = No parity
 1 = Even parity
 2 = No parity
 3 = Odd parity
- NN = 0 = illegal value
 1 = 1 stop bit
 2 = 1.5 stop bits
 3 = 2 stop bits

SSSS = 0 = 110 baud
1 = 134.5 baud
2 = 150 baud
3 = 200 baud
4 = 300 baud
5 = 600 baud
6 = 1200 baud
7 = 1800 baud
8 = 2400 baud
9 = 4800 baud
A = 9600 baud
B-F = illegal values

MM = 0 = Half duplex operation
1 = Half duplex operation with automatic insertion
and removal of shift characters
2 = Echoplex operation
3 = Full duplex operation

X = 0 = No receive data synchronization
1 = Synchronize received data to begin with
any of the 4 synchronization characters
set in the CCV.

C = 0 = No code translation performed by
controller
1 = Transmit and receive code translation
performed by controller.

MODEL 2227B

BUFFERED ASYNCHRONOUS COMMUNICATIONS CONTROLLER

INTERIM MANUAL

(JUNE 7, 1976)

Disclaimer of Warranties and Limitation of Liabilities

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

WANG

LABORATORIES, INC.

836 NORTH STREET, TEWKSBURY, MASSACHUSETTS 01876, TEL. (617) 851-4111, TWX 710 343-6769, TELEX 94.7421

CONTENTS

Page

CHAPTER 1	MODEL 2227B FEATURES	
1.1	General Information	1
1.2	Installation	3
1.3	Connector Pin Assignments	4
1.4	Controller and Modem Interaction	5
1.5	Asynchronous Transmission/Reception	6
	Character Transmission	7
	Character Reception	7
1.6	Data Buffering	8
1.7	Substitution for Characters Received in Error	9
1.8	Transmission Delays Following Specified Characters	9
1.9	Code Translation	9
1.10	Insertion and Removal of Shift Characters	10
1.11	Detecting End of Record Characters	11
1.12	Echoplex Operation	12
1.13	Monitoring Received Timeouts	12
1.14	Sending and Detecting Break Signals	12
CHAPTER 2	PROGRAMMING TECHNIQUES	
2.1	General Considerations	14
2.2	Specifying the Communications Control Vector	15
2.3	The Communications Status Vector	19
2.4	CPU and Controller Interaction Via \$GIO Statements	20

CHAPTER 1
MODEL 2227B FEATURES

1.1 GENERAL INFORMATION

Physically the Model 2227B Buffered Asynchronous Communications Controller is a double-card controller which plugs into any I/O slot in a System 2200 Central Processing Unit (CPU). For program control of data transmission and reception via the controller, the CPU must include the \$GIO statement in its BASIC language instruction set.

A 25-pin EIA (Electronic Industries Association) RS-232-C, CCITT V.24 compatible connector mounted on the controller and a Wang-supplied cable (length 12 feet, i.e., 3.6 meters) facilitate hookup of a modem for communications applications. No operating controls are mounted on the controller.

Though primarily designed for communications applications, the Model 2227B controller is well-suited for direct connection of RS-232-C compatible asynchronous transmission equipment such as the following: local CRT terminals, graphic display terminals, analytical equipment and laboratory instrumentation in general. For hookup of such equipment, a null modem (available from Wang Laboratories) may be required.

The controller has an integrated microprocessor and multicharacter input/output buffers to simplify telecommunications control procedures and reduce CPU processing requirements. Fixed microcode, residing in the controller's read-only-memory, implements standard and optional operations such as the following:

- . data buffering
- . code translation
- . substitution for characters received in error
- . communications control, e.g., monitoring CPU ready/busy conditions, monitoring modem signals, implementing line turnaround procedures
- . break signal detection and transmission
- . detection of received timeouts
- . detection of end of record characters
- . automatic insertion and removal of shift characters
- . automatic transmission delays following several specified characters

- automatic echoplex operation
- sensing the Secondary Received Line Signal Detector and setting the Secondary Request to Send signal (also called reverse or supervisory channel data signals).

The standard and special features of the controller are designed to enable a System 2200 to be programmed to transmit and receive data using the line discipline of a variety of asynchronous CRT and mechanical printer terminals.

The controller's 1K-byte random-access-memory is used for storage of initialization information (including code translation tables and a communications control vector), 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 conveniently selected under program control by specified values in a communications control vector. The vector is loaded into the controller by a \$GIO statement in the user's communications program operating in the CPU.

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

The Model 2227B controller supports any one of the following transmission modes:

- half-duplex operation (independent transmission one-way-at-a-time alternately)
- half-duplex operation with automatic deletion of null characters
- full-duplex operation (independent transmission two-ways simultaneously)
- echoplex operation (a type of error checking, requiring immediate retransmission of received characters while operating in a full-duplex mode).

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

Via other byte-positions in the communications control vector, the character format can be selected from the following options:

- 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 features of the Model 2227B Buffered Asynchronous Communications Controller are numerous; descriptions of particular features are presented in this chapter. Programming techniques are presented in Chapter 2, where the format of the communications control vector is shown byte-by-byte in Figure 2-1 and valid values representing the asynchronous transmission options are presented in Table 2-1; also, the format of the communications status vector is given, and \$GIO microcommand sequences needed to operate the controller are supplied.

1.2 INSTALLATION

Installation of a Model 2227B controller is the responsibility of a Wang Service Representative who should be called when the controller arrives.

After the controller is inspected, diagnostically checked, and installed in an I/O slot in a System 2200 CPU, one end of the cable supplied with the controller is plugged into the connector located on the controller faceplate. The other end of the cable has an RS-232-C compatible male plug.

If the Model 2227B controller is to be used for point-to-point, dial-up asynchronous telecommunications applications, the other end of the cable should be plugged into a suitable modem. Installation of a modem is not the responsibility of a Wang Service Representative.

If the Model 2227B controller is to be used to interface RS-232-C compatible asynchronous transmission equipment such as a non-Wang CRT terminal or a laboratory instrument, the other end of the cable may need to be plugged into a null modem (available from Wang Laboratories) or may be suitable for direct connection to the non-Wang equipment. Installation or interfacing of non-Wang equipment is not the responsibility of a Wang Service Representative; therefore, information regarding the connector pin assignments and voltage levels is given in Section 1.3.

Modem Considerations

The modem used with the Model 2227B controller may be rented from the telephone company serving the locality where a Wang system is installed or may be purchased from any one of several modem vendors. In either case, arrangements should be made with the telephone company for installation of a modem since modems purchased from a vendor must be connected to the telephone network via telephone company installed data access arrangements (DAA) consisting of a telephone handset and modem interface rented from the telephone company.

Before a telephone company representative arrives to install a modem, the location of the System 2200 must be planned to ensure its proximity to the telephone equipment. Normally modems or DAA's are wired permanently to a wall; in such cases, subsequent relocation of the System 2200 any great distance would necessitate having the telephone company relocate the modem or DAA. 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.

Other modem considerations are discussed in Section 1.4.

1.3 CONNECTOR PIN ASSIGNMENTS

The Model 2227B controller 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:

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

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

Table 1-1. Model 2227B 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 Detector	Modem
9				
10				
11	SCA	120	Secondary Request to Send	Controller
12	SCF	122	Secondary Rec'd Line Sig. Det.	Modem
13*	SCB	121	Secondary Clear to Send	Modem
14*	SBA	118	Secondary Transmitted Data	Controller
15*	DB	114	Trans. Signal Element Timing	Modem
16*	SBB	119	Secondary Received Data	Modem
17*	DD	115	Receiver Signal Element Timing	Modem
18*		124	Select Frequency Groups	Controller
19*	SCA	120	Secondary Request to Send	Controller
20	CD	108.2	Data Terminal Ready	Controller
21*	CG	110	Signal Quality Detector	Modem
22*	CE	125	Ring Indicator	Modem
23*	CH/CI	111/112	Data Signalling Rate Selector	Controller/Modem
24*	DA	113	Trans. Signal Element Timing	Controller
25			Unassigned	

* Signals not utilized by controller.

1.4 CONTROLLER AND MODEM INTERACTION

The microprocessor on the Model 2227B controller 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
- . Secondary Received Line Signal Detector on Pin 12.

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
- . Secondary Request to Send on Pins 11 and 19.

Any Bell 103 or 202 type modem, or equivalent, can be used with the controller. Very likely, other RS-232-C compatible modems commonly used with asynchronous terminals having transmission rates in the range covered by the Model 2227B controller will prove suitable.

Normally, a 103A type modem is used for operations requiring line transmission speeds up to 300 bits per second. The modem should be ordered with optional features which provide an originate and an answer capability. Also, a receive long space disconnect feature is desirable for use with the Model 2227B controller, but no special features are necessary for a break signal capability.

Normally, a 202C or 202S type modem is used for operations requiring line transmission speeds at 1200 bits per second (and occasionally at lower speeds): Options should include originate, answer, and receive long space disconnect features. Furthermore, if use of a break signal is desired (as is the case more often than not), the reverse channel option should be included.

Note:

1. Modems used at both ends of a point-to-point dial-up communications link must be of similar type. For example, if a Bell 103 type modem is used at one end, another Bell 103 type modem or an equivalent modem must be used at the other end (not a Bell 202 type modem).
2. 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.

1.5 ASYNCHRONOUS TRANSMISSION AND RECEPTION

In asynchronous transmission, often called start-stop 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.

Depending upon the design of the transmitting equipment, the data-bit interval is a single 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 single value or one of several possible values, depending upon the equipment being 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 intervals. The voltage level may be a logic "0" or "1" depending upon the type of parity (odd or even) as well as the number of 1's 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. Generally speaking, there is no upper limit to the length of a stop element; however, there is a lower limit depending upon the design of the transmitting equipment. The lower limit is a single value or one of several possible values.

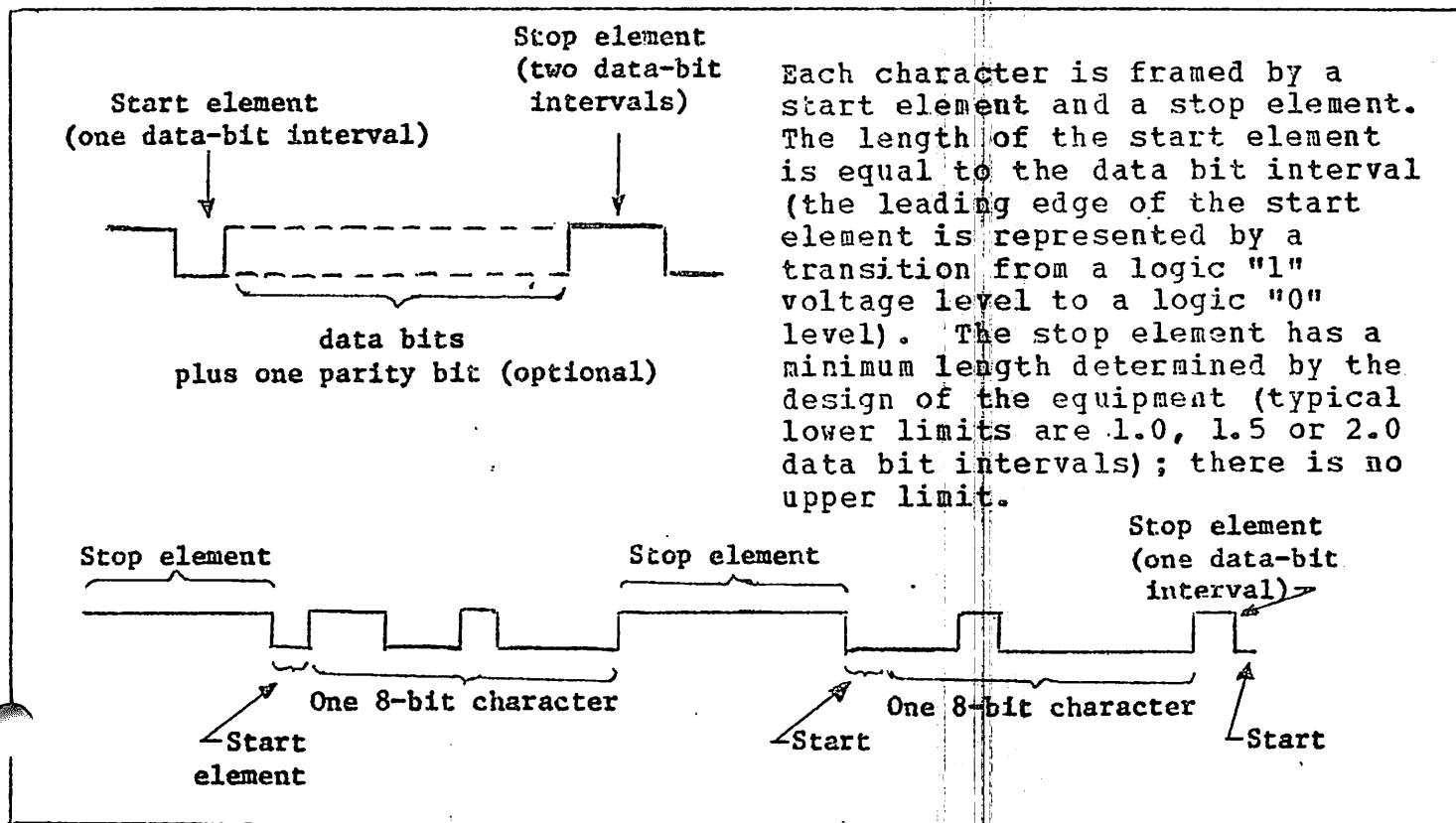


Figure 1-1. Asynchronous Data Transmission

Character Transmission

The Model 2227B controller transmits each character by modulating the Transmitted Data signal on Pin 2 in the connector as follows:

1. The Transmitted Data signal is set to "0" for one bit-time, representing the start bit.
2. Successively, low-order bit first, the signal is set for one bit-time to the value of each data bit until the number of transmitted data bits equals the number specified in the communications control vector; therefore, only the low-order 5, 6, 7 or 8 bits of a character are transmitted.
3. If odd or even parity is specified in the communications control vector, the signal is set for one bit-time to the appropriate value for the type of parity specified. In particular, if odd parity is specified, the parity bit is equal to 1 when the preceding data bits contain an even number of 1-bits (i.e., for odd parity, the total number of 1's in the data bits plus the parity bit is an odd number). If even parity is specified, the parity bit is equal to 1 when the preceding data bits contain an odd number of 1-bits (i.e., for even parity, the total number of 1's in the data bits plus the parity bit is an even number). If no parity is specified, Step 3 is omitted.
4. The Transmitted Data signal is set to "1" for a minimum interval equal to 1, 1.5 or 2 bit-times depending upon the number of stop bits specified in the communications control vector.

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

Character Reception

The Model 2227B controller receives a character by detecting changes in the Received Data signal on Pin 3 in the connector as follows:

1. A transition from the voltage level representing logic "1" to the level representing logic "0" for at least one-half a bit-time is interpreted as the leading edge of the start bit for an incoming character.
2. The Received Data signal is sampled successively at times corresponding to the nominal center of each data bit. In particular, the nominal center of the first data bit is 1.5 bit-times after the leading edge of the start bit. The center of each subsequent bit occurs one bit-time after the center of its predecessor. Successively, low-order bit first, the bits in the character being receiving are set to correspond to the sampled values. The number of data bit samples taken by the controller equals the number of data bits specified in the communications control vector. If the number of samples is less than 8, the remaining high-order bits in the received character are automatically set to 0 (unless the shift character option is in effect).

3. A parity bit, if specified, is read by sampling the Received Data signal again -- one bit-time after the last data-bit is sampled. The sampled parity value is compared with a calculated value based on the received data bits and the type of parity specified. If the received and calculated parity values are unequal, a designated bit in the communications status vector is set to 1 to indicate a parity error has occurred.
4. One bit-time after the Received Data signal is sampled for a parity value (or 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 chosen for the Model 227B controller must match the specifications for equipment in use at the other end of a communications link.

1.6 DATA BUFFERING

The controller has two multicharacter data buffers, a 200-byte transmit buffer and a 255-byte receive buffer. With these buffers, data transmission/reception operations performed by the controller with respect to the modem can overlap data input/output operations performed by the CPU with respect to the I/O peripherals designated for a communications application.

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

Note:

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

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

Note:

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

1.7 SUBSTITUTION FOR CHARACTERS RECEIVED IN ERROR

When a character is received with either a parity or a framing error, a substitute character (defined by byte 4 in the communications control vector) is placed in the controller's receive buffer, and the appropriate error bit is set in the communications status vector. For example, a special character such as @, or any other character not likely to occur in the incoming data, can be specified as the character to automatically replace any characters received in error.

1.8 TRANSMISSION DELAYS FOLLOWING SPECIFIED CHARACTERS

When sending data to a mechanical printer terminal such as an IBM 2741 or a Wang 1200, time delays are needed after transmission of TAB and CR (carriage return) characters to allow the printing element sufficient time to reach the proper position without loss of subsequent characters.

A special feature of the Model 2227B controller removes the necessity for the user's program operating in the CPU to introduce time delays following transmission of special characters. Bytes 11 through 18 of the communications control vector can be used, in four two-byte groups, to define up to four special characters and the transmission delay to follow each special character. During transmission, the controller automatically delays for the specified time following the transmission of one of the specified characters.

1.9 CODE TRANSLATION

The code translation feature of the Model 2227B controller allows data interchange between the CPU and the controller in the ASCII code (American Standard Code for Information Interchange) native to the System 2200 -- regardless of the transmission/reception code for a particular application.

Space is reserved in the controller 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 program operating in the CPU, are loaded into the controller by \$GIO statements in the program.

The automatic code translation operation is enabled by loading the transmit and receive code translation tables 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 controller 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 communications control vector specifies less than 8 data bits per character, only the relevant low-order bits of each character are transmitted.

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

Note:

Superfluous characters used for timing or fill can be removed automatically by translating them to null characters. This feature is applicable to half-duplex operations. See Table 2-1.

1.10 INSERTION AND REMOVAL OF SHIFT CHARACTERS

For applications involving data transmission and reception in a code set which utilizes shift characters, e.g., when emulating an IBM 2741 terminal or a Wang 1200 terminal, a special feature of the Model 2227B controller removes the necessity for the user's program operating in the CPU to handle shift characters.

The upshift and the downshift characters are defined in bytes 7 and 8 of the communications control vector. To activate automatic insertion and removal of shift characters, code translation tables must be loaded into the controller, and the number of data bits per character in the transmitted and received data must be 5 or 6.

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

<u>Two High-order Bits</u>	<u>Meaning</u>
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 particular, an upshift character is

transmitted prior to a character having upshifted status, and a downshift character prior to a character having downshifted status. The shift-status bits are not transmitted since the number of data bits transmitted per character should be only the low-order 5 or 6 bits.

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

<u>High-order Bit</u>	<u>Meaning</u>
1	Upshifted character.
0	Downshifted character.

Note:

If a received character is a shift character, the character is discarded; otherwise, the character is code translated and placed in the receive buffer.

1.11 DETECTING END OF RECORD CHARACTERS

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

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

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

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

With a \$GIO statement described in Section 2.4, the user's communications program can read the status vector into the CPU for testing to ensure the availability of a complete record before requesting transfer of buffered data via another \$GIO statement.

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

If the end-of-record detection feature is not needed for an application (can not be utilized because the high-order bit for codes in the receive-code translation table are set to 1 for a purpose other than defining an end-of-record character), byte 6 in the communications control vector should be set to HEX(C0) to disable end-of-record detection.

1.12 ECHOPLEX OPERATION

Echoplex operation is a type of error checking used by some computer systems in their support of Teletype[®] terminals. Both the modems and the transmission facilities must be full-duplex. The terminal is expected to immediately retransmit each received character, so the computer can compare the retransmitted character with the one originally transmitted.

To activate the echoplex mode of operation, the high-order hexdigit for byte-position-2 in the communications control vector must be set to the value 2. In the echoplex mode, the Model 2227B controller operates as if in the full-duplex mode with one exception -- received characters are retransmitted (echoed) automatically.

Note:

If a character is received with a parity or a framing error, the substitute character defined in byte 4 of the communications control vector is echoed.

1.13 MONITORING RECEIVED TIMEOUTS

The Model 2227B controller has the capability to monitor received timeouts. Byte 5 in the communications control vector is used to set the binary value of the timeout in units of 0.1 seconds. For example, the minimum timeout condition 0.1 seconds is specified by storing HEX(01) in byte-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.

Note:



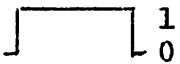
If no valid characters are received during a specified period of time after a "start receiving data" \$GIO operation, the controller sets a particular bit in the status vector. No other action is taken by the controller when a timeout condition is exceeded.

1.14 SENDING AND DETECTING BREAK SIGNALS

The Model 2227B controller has the capability to send and detect break signals under control via the user's program operating in the CPU. Bytes 9 and 10 in the communications control vector 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(10) stored in byte-position-10 defines an interval equal to 160 milliseconds for detection of break signals.

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

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

Note:

1. The Transmitted Data and Received Data modem signals are used with Bell 103 type modems.
2. Normally the Secondary Request to Send and Secondary Received Line Signal Detector modem signals are used with Bell 202 type modems which must be ordered with the reverse channel option in order to support break signal operation.

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

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

Note:

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

CHAPTER 2 PROGRAMMING TECHNIQUES

2.1 GENERAL CONSIDERATIONS

When writing user programs for the Model 2227B Buffered Asynchronous Communications Controller, the programmer should organize each applications program in distinct modules for initialization and communications functions. As preferred, the communications module could overlay the initialization module, or the two modules could coexist simultaneously.

Generally speaking, an initialization module defines the 20-byte communications control vector and assigns values representing 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 controller is to activate such features as break detection, monitoring of received timeouts, half or full duplex operation, 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 controller.

If required, an initialization module defines transmit and receive code translation tables, and supplies the \$GIO statements needed to load such tables into the controller. Furthermore, the module might define and initialize variables required by the communications module even though the variables are stored in the CPU rather than the controller memory.

2.2 SPECIFYING THE COMMUNICATIONS CONTROL VECTOR

The format of the communications control vector is shown in Figure 2-1 and valid specifications for the first three bytes of the vector are given in Table 2-1.

In the user's communications program residing in the CPU, the control vector should be defined by a one-dimensional array having 18 or 20 elements with one byte per element, e.g.,

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

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

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

before assigning values to particular elements.

Individual byte-positions in the control vector can be assigned values other than binary zero to select a desired combination of options and define special characters for a variety of operations as illustrated by the following statements.

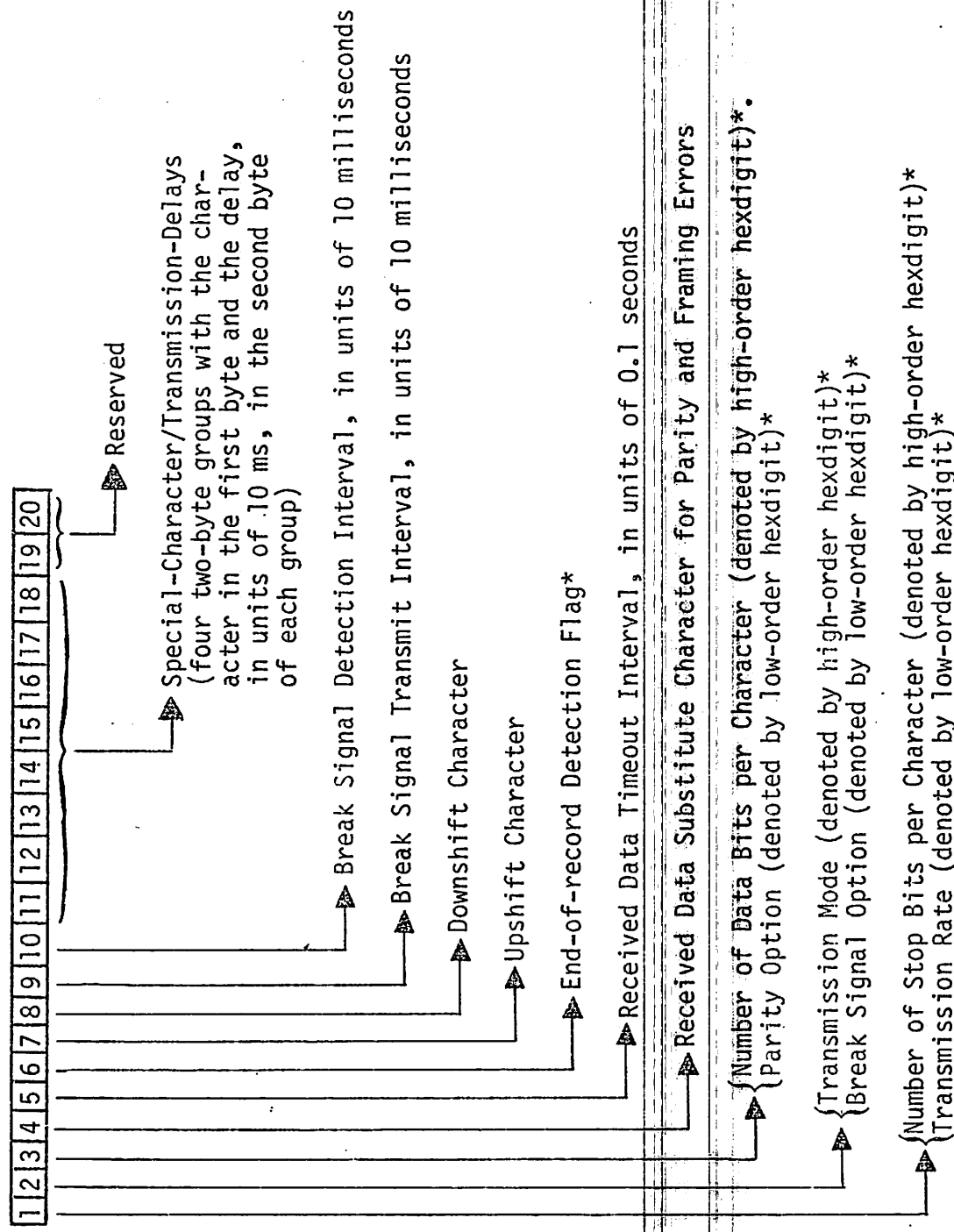
Statement

Comment

```
C$(1) = HEX(17)    One stop bit; 300 bits per second.  
C$(3) = HEX(23)    Seven data bits; odd parity.
```

Care must be exercised when defining some special characters to choose a compatible value in the first three byte positions of the communications control vector. For example, if defining upshift and downshift characters in bytes 7 and 8, the high-order hexdigit in byte 3 must have the value 0 or 1 (since the shift feature can be used only with code sets having 5 or 6 data bits per character). See Section 1.10 and Table 2-1.

COMMUNICATIONS CONTROL VECTOR BYTE POSITIONS



*See Table 2-1.

Figure 2-1. Communications Control Vector Format

Table 2-1. Valid Communications Control Vector Specifications

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

*See Table 2-1 (Continued) for bytes 4 through 20.

Table 2-1. Valid Communications Control Vector Specifications (Continued)

Byte	High and Low Order Hexdigits*	Remarks
4	xy = Substitute character for parity/framing errors.	Each received character having a parity or framing error is replaced by the designated character (replacement occurs prior to code translation if translation tables are being used). See Section 1.7.
5	xy = Timeout interval in units of 0.1 seconds.	The specification in hexadecimal notation represents the timeout interval in units of 0.1 seconds, e.g., $(24)_{16} = (36)_{10}$ specifies an interval of 3.6 seconds. See Section 1.13.
6	00 = Disable end-of-record detection. 01 = Enable end-of-record detection.	If enabled, the end-of-record characters must be defined via the receive-code translation table by setting the high-order bit to 1 for each code corresponding to an incoming end-of-record character. See Section 1.11.
7	xy = Upshift character.	To enable shift code insertion/deletion, the high-order hexdigit in byte 3 of the control vector must be 0 or 1 (i.e., the number of data bits per character must be 5 or 6). Also, the transmit-code translation table must identify all downshifted, upshifted, and "don't care" characters by setting the two high-order bits to 00, 01, and 10 or 11 as described in Section 1.10. The receive-code translation table must allow for the controller's automatic setting (before translation) of the high-order bit to 1 for all incoming upshifted characters.
8	xy = Downshift character.	
9	xy = Break signal transmit interval in units of 10 ms.	To enable break signal transmission/detection, the low-order hexdigit in byte 2 of the control vector must specify the polarity and the modem signals. If bytes 9 and 10 are both HEX(00), the low-order hexdigit in byte 2 should be 0. The byte 9 and 10 specifications in hexadecimal notation represent break signal transmit and receive intervals in units of 10 milliseconds, e.g., $(12)_{16} = (18)_{10}$ specifies a 180 ms interval. See Section 1.14.
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 is HEX(FF)~2.5 seconds. During transmission, the controller automatically delays for the specified time following the 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	00 = No transmission delay. 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.	

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

The communications control vector should be loaded into the controller by a \$GIO statement having the appropriate microcommand sequence from Table 2-3. See Section 2.4.

2.3 THE COMMUNICATIONS STATUS VECTOR

Space is reserved in the random access memory of the Model 2227B controller for a communications status vector whose byte and bit positions are used automatically as shown in Table 2-2. The first three bytes of the status vector are cleared automatically whenever it is read and when the communications control vector is loaded into the controller from the CPU. See Section 2.4.

Flags are set in the first three bytes of the status vector during controller 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. Byte 6, on the other hand, is similar to a real time clock whose value is initiated to the timeout interval specified in byte 5 of the communications control vector each time one of the following events occurs:

- a) A \$GIO "start receiving data" operation begins,
- b) a line turnaround occurs during a \$GIO "send, then receive data" operation, or
- c) a character is received during either operation.

If the value in byte 6 of the status vector is not reset by one of these operations, the countdown proceeds to zero.

Whenever desired, the information in the status vector can be read (transferred to the CPU) by a \$GIO statement having the appropriate microcommand sequence from Table 2-3. See Section 2.4. After transfer to the CPU, status vector information can be tested, as required.

Table 2-2. Communications Status Vector Information

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

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

2.4 CPU AND CONTROLLER INTERACTION VIA \$GIO STATEMENTS

To operate the Model 2227B controller, the user's communications program residing in the CPU should include \$GIO statements with suitable microcommand sequences. A list of valid microcommand sequences is presented in Table 2-3 for several operations.

Table 2-3. Valid Microcommand Sequences for Model 2227B Operations

Operation	Microcommand Sequence*
Set communications control vector	4580 4402 A000 440C
Read communications status vector	4403 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
Transfer received data to CPU	4409 C620
Send data	440A A000 440C
Send, then receive data	440B A000 440C

*A microcommand sequence can be specified directly or indirectly in a \$GIO statement. If specified directly as the arg-1 component, each four-hexdigit-code can be separated from the previous one by a space for readability as shown in this table. If specified indirectly by assigning the sequence to a variable and including the variable in a statement, spaces cannot be used, e.g., A\$ = HEX(44804402A000440C). See the General I/O Instruction Set Reference Manual.

Brief descriptions of each of the operations in Table 2-3 follow. A sample \$GIO statement is shown for each operation; however, the comments and variables used in the statement may be given different names in a user's program, and the arg-1 variable can be eliminated if the microcommand sequence is specified in the statement.

Set Communications Control Vector

```
$GIO SET CCV /01C (G0$, G$) C$()
```

The communications control vector (CCV) defined by the array C\$() is set (loaded) into the controller when the statement is executed. Here, 01C is the address of the controller and G0\$ represents the first microcommand sequence shown in Table 2-3. G\$ represents the error/status/general-purpose registers.

Note:

The controller transmit and receive buffers, as well as the communications status vector and code translation tables, are cleared.

Read Communications Status Vector

\$GIO READ CSV /01C (G1\$, G\$) A\$

The character string A\$ (which must be at least 6 bytes long) is set to the six-byte value of the communications status vector. Here G1\$ represents the second microcommand sequence from Table 2-3.

Note:

The error and received break indicators in the communications status vector are cleared.

Load Transmit Code Translation Table

\$GIO LOAD TTBL /01C (G2\$, G\$) C1\$()

The transmit code translation table (which must be defined in the user's program) is loaded into the controller from the array C1\$(). However, execution of code translation by the controller is optional. The feature is enabled if transmit and receive code translation tables are loaded after the communications control vector is loaded into the controller. Here, G2\$ represents the third microcommand sequence from Table 2-3. The array C1\$() must be previously defined in the user's communications program.

NOTE:

1. The transmit translation table should be exactly 256 bytes in length and represent the codes to which System 2200 ASCII characters are to be converted prior to transmission. The byte positions in the table should contain the "after translation characters" arranged in a sequence corresponding to the "before translation characters", i.e., the System 2200 ASCII characters.
2. If shift character automatic insertion/removal is in effect (i.e., the specified number of data bits per character is 5 or 6), the high-order two bits of each code in the transmit translation table must conform to the appropriate values given in Section 1.10.

Load Receive Code Translation Table

SGIO LOAD RTBL /OIC (G3\$, G\$) C2\$()

The receive code translation table is loaded into the controller from the array C2\$() which must be previously defined. Here, G3\$ represents the fourth microcommand sequence from Table 2-3. Execution of code translation is optional, depending upon whether code translation tables are loaded into the controller by the user's communications program.

NOTE:

1. The receive 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. If the incoming non-ASCII character is a HEX(18), the binary value is 24; therefore, the corresponding ASCII character should be located in the 25th position of the receive translation table. (Keep in mind that the first position corresponds to the binary value zero.)
2. If shift character automatic insertion/removal is in effect, the 256-byte receive translation table represents two 128-byte tables. The first 128 bytes in the table should represent the conversions for incoming downshifted characters corresponding to the hexadecimal codes HEX(00) through HEX(7F). The second 128 byte positions should represent conversions for incoming upshifted characters corresponding to the hexadecimal codes HEX(80) through HEX(FF).
3. If end-of-record character detection is enabled, the high-order bit for codes in the receive-code translation table must be set to 1 for each character defined as an end-of-record character (and set to zero for other characters).

Disconnect

```
$GIO DISCONNECT /01C (G4$, G$)
```

If G4\$ represents the microcommand 4406 shown in Table 2-3, the controller disconnects from the line by setting the Data Terminal Ready signal to zero for a period of three seconds.

Send Break

```
$GIO BREAK /01C (G5$, G$)
```

If G5\$ represents the microcommand 4407 shown in Table 2-3, the controller sends a break signal in accordance with the circuit and polarity denoted by the low-order hexdigit in byte-position-2 of the communications control vector. See Table 2-1 and Section 1.14.

Start Receiving Data

```
$GIO START RCV /01C (G6$, G$)
```

If set for half-duplex operation, the transmit and receive buffers are cleared, and the controller enters the receive mode and starts receiving data. Also, the receive timeout countdown is started, and its current value is maintained in byte 6 of the communications status vector. Here, G6\$ represents the microcommand 4408 shown in Table 2-3.

Transfer Received Data to the CPU

```
$GIO RCV /01C (G7$, G$) D$()
```

The contents of the receive buffer are transferred from the controller to the CPU and stored in the array D\$(). 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 received whether stored or not. See the General I/O Instruction Set Reference Manual. Here, G7\$ represents the eighth microcommand sequence shown in Table 2-3. See Section 1.11. Note: The \$GIO statement buffer, here represented by D\$(), should be an array at least 255 bytes long.

Send Data

```
$GIO SEND /01C (G8$, G$) F$() <1, N>
```

If set for half-duplex operation, the receive buffer is cleared. The first N bytes of the array F\$() are transferred from the CPU to the controller for storage in the transmit buffer. The controller transmits the data and remains in the transmit mode (if set for half-duplex operation). Here, G8\$ represents the ninth microcommand sequence in Table 2-3.

Send Then Receive Data

```
$GIO SEND RCV /01C (G9$, G$) F$() < 1, N >
```

This statement is applicable only for half-duplex operation. The first N bytes of the array F\$() are transferred from the CPU to the controller for storage in the transmit buffer. The controller transmits the data and then executes a "start receiving data" operation. Here, G9\$ represents the tenth microcommand sequence in Table 2-3.

NOTE:

1. For half-duplex communications, the "send data" \$GIO operation should be used to send all but the last bytes of data. The "send, then receive data" operation should be used to send the last bytes of data, and afterwards the "transfer received data to CPU" should be used. Use of the "send, then receive data" operation automatically implements a line turnaround procedure, thereby ensuring the controller's readiness to receive data without loss.
2. For full-duplex communications, only the "send data" and "transfer received data to CPU" are needed. The "send, then receive data" operation should not be used.

- 9) Pin 15 TRANSMITTER SIGNAL ELEMENT TIMING (FROM data set).

The square wave signals on this circuit (2000 HZ for 2000 baud modem; 2400 HZ for 2400 baud modem; 4800 HZ for 4800 baud modem) are used to provide the data terminal equipment with signal element timing information for the transmitted data circuit. A timing signal will be present on this circuit whenever power is on in the data set.

- 10) Pin 17 RECEIVER SIGNAL ELEMENT TIMING (FROM data set).

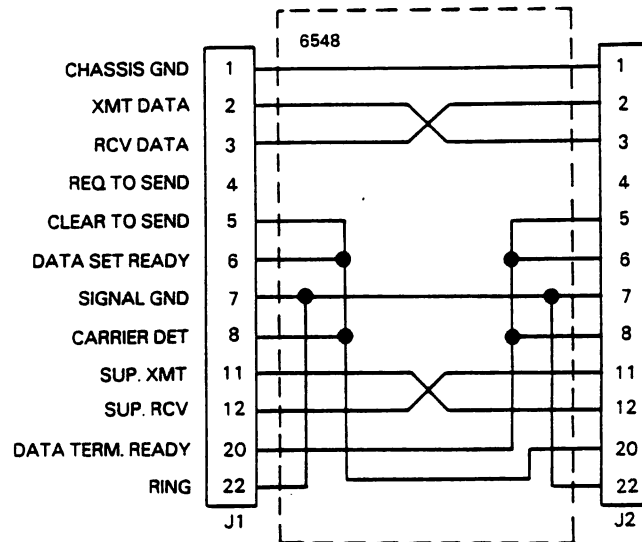
The square wave signals on this circuit (2000 HZ for 2000 baud modem; 2400 HZ for 2400 baud modem; 4800 HZ for 4800 baud modem) are used to provide the data terminal equipment with receiver signal element timing information. The transition from ON to OFF normally indicates the center of each signal element on the received data circuit. A timing signal will be present on this circuit when CARRIER (pin 8) is ON for data sets 201A and 201C.

- 11) Pin 20 DATA TERMINAL READY (TO data set).

The data terminal must apply an ON condition (+3v to +25v) to this circuit at all times to go into the Data mode. An OFF signal on this line will not allow data set to go to the Data mode. An OFF signal on this line during data reception or transmission will make the data set drop the communication line.

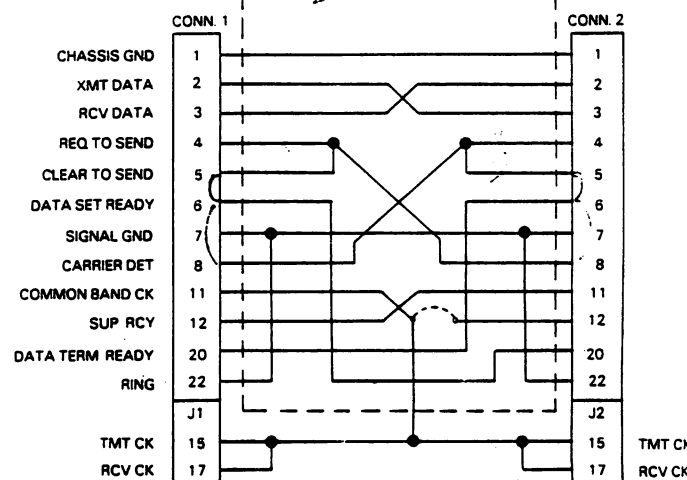
2227N NULL MODEM

ASYNC



2228N NULL MODEM

31 SYNC



```

199 * THE COMMUNICATIONS CONTROL VECTOR IS AS FOLLOWS:
200 *
201 *     BYTE 1     - - - C - - - X
202 *
203 *     C:      1= PERFORM CODE TRANSLATION
204 *     X:      1= RECEIVE CHAR SYNC. ENABLED
205 *
206 *     BYTE 2     - - N N S S S S
207 *
208 *     NN:     0,1= 1 STOP BIT
209 *             2  1.5 STOP BITS
210 *             3  2 STOP BITS
211 *
212 *     SSSS:   0-3= 50, 75, 100, 110 BAUD
213 *             4-7 134, 150, 200, 300 BAUD
214 *             8-B 600, 1200, 1800, 2400 BAUD
215 *             C-F 3600, 4800, 7200, 9600 BAUD
216 *
217 *     BYTE 3     - - M M - - B B
218 *
219 *     MM:     0= HALF DUPLEX
220 *             1  HALF DUPLEX WITH SHIFT CODES
221 *             2  ECHOPLEX
222 *             3  FULL DUPLEX
223 *
224 *     BB:     0= BREAK DISABLED
225 *             1  BREAK ENABLED ON TRANSMIT/RECEIVE
226 *             2  BREAK ENABLED ON SECONDARY REQ.
227 *             TO SEND & SEC. REC. LINE SIG. DET
228 *             3  SAME AS 2 WITH INVERTED POLARITY
229 *
230 *     BYTE 4     - - W W - - P P
231 *
232 *     WW:     0-3= 5, 6, 7, 8 BIT DATA WORD
233 *
234 *     PP:     0= NO PARITY
235 *             1  EVEN PARITY
236 *             2  NO PARITY
237 *             3  ODD PARITY
238 *
239 *     BYTE 5     RECEIVED ERROR SUBSTITUTE CHARACTER
240 *     BYTE 6     RECEIVE TIMEOUT VALUE TIMES 0.1 SECON
241 *     BYTES 7 -10 RECEIVE SYNCHRONIZATION CHARACTERS
242 *     BYTES 11-14 RECEIVE END-OF-RECORD CHARACTERS
243 *     BYTE 15     UP-SHIFT CHARACTER
244 *     BYTE 16     DOWN-SHIFT CHARACTER
245 *     BYTE 17     BREAK TRANSMIT TIME TIMES 0.01 SECOND
246 *     BYTE 18     BREAK DETECT TIME TIMES 0.01 SECONDS
247 *     BYTE 19     INSERTED TRANSMIT IDLE CHARACTER
248 *     BYTES 19-28 4 PAIRS OF THE FOLLOWING:
249 *             CHAR TO BE FOLLOWED BY TR. IDLE
250 *             NUMBER OF IDLE CHARACTERS

```

