# A BLOCK ORIENTED INTERFACE
# FOR CP/M* AND THE VADCG TERMINAL NODE CONTROLLER

Douglas Lockhart, VE7APU/3
29 Shamokin Drive
Don Mills, Ontario M3A 3H7
416-441-2417

## Abstract

This paper describes a system of hardware and software which provides for the transfer of blocks of data between a VADCG Terminal Node Controller (TNC) and a CP/M system with a serial interface. Both the software to run in the TNC and in the CP/M system is included. The system provides block transfers, data transparency, flow control and error checking and retransmission in both directions over the interface.

## Introduction

The software to implement the Link level of protocol for the VADCG Terminal Node Controller was developed in 1978. It is now in general use both in the U.S. and Canada and has even been implemented on other Terminal Node Controller boards. It has proven to be satisfactory for the purposes intended but many people recognize the need to implement the next higher level of protocol - the Packet or Network level protocol.

There have been a large number of proposals as to the form this protocol should take and I have made my own proposals in a paper published in the last Amateur Radio Computer Networking Conference. In spite of a large supply of proposals there is a distinct shortage of implementations. Part of the reason for this has been because of the need for some kind of consensus in the Amateur Radio fraternity. Notwithstanding this important concern, there is another reason why we don't have our Network level protocol implemented - it is a lot of work to get it going.

'What are the problems in implementing the Network level protocol?', you may ask. Well, unlike the Link level protocol which only had to be implemented to run in a TNC, parts of the Network level protocol have to be implemented to run in each microcomputer connected to the network. Furthermore, the TIP programs in the TNCs will have to be rewritten and some changes in the LIP programs are needed as well. In addition, the Network level protocol is much more complex than the link level protocol. I think one of the main stumbling blocks is the need to implement the protocol on two separate systems before any testing can be done.

Despite the above difficulties, I have begun the process of implementing the protocol and have broken the job down into steps that can be implemented and tested and then proceed to the next step. To alleviate the problem of having to make two implementations for different systems, I am only making one implementation for my CP/M system which I will hopefully be able to transport to another local Packeteer's CP/M system for testing. In order to make this program as transportable as possible to other CP/M systems I am only using the 8080 instruction set.

The programs here are not really any part of a higher level protocol but the function they perform will be needed by any higher level protocol that is adopted. The microcomputer program called 'PACKET' is basically a set of drivers for the serial interface between the microcomputer and the TNC. The program implementing the higher level of protocol in the microcomputer is called the Transmission Control Program or TCP. The TCP will use these drivers to transfer blocks of data that it has prepared to the TNC and it will also receive blocks of data from the TNC using these drivers.

The TCP is called upon by the programs running in the microcomputer to send data and receive data to and from various points in the network. In order to do this job, the TCP adds a header onto the outgoing blocks of data and because the bits and bytes in this header have a meaning based on their position in the block of data, there must be a mechanism to show where a block starts and ends in the serial data streams being passed accross the interface between the computer and the TNC. This mechanism, was lacking in all the TIPs that I had access to. Also, since flexibility in the setting of these bits was needed and any 'kind of restriction on the data being sent across the interface was undesirable, there had to be a mechanism for data transparency. This mechanism, too, was missing in all the TIPs that I had access to. Also, since data was being sent both ways at high speed by microprocessors, there had to be a mechanism for flow control in both directions across the interface. Also, since my serial interface used long RS-232 cables in a noisy environment, I occasionally got bit errors in the data especially at the higher speeds so I needed to have error detection in this interface. In some environments, error detection may not be necessary but I decided to play it safe and include it. Finally, error detection is not of much use unless you can correct the errors so I have incorporated a retransmission mechanism.

To summarize - the interface provides the following:

1. Block recognition.
2. Data transparency.
3. Flow control (in both directions)
4. Error detection.
5. Error correction.

A block has the following format:

| SYN | DLE | STX | DATA | DLE | ETX | CRC | PAD |
|---|---|---|---|---|---|---|---|
| 16H | 10H | 02H |  | 10H | 03H |  | FFH |

The combination DLE-STX (ASCII Data Link Escape and Start of Text) indicates the start of a transparent block of data and the combination DLE-ETX indicates the end of the transparent block. To provide for data transparency a 'byte stuffing' technique is used - any time transparent data occurs that looks like a DLE, then an extra DLE is stuffed into the data stream. Therefore, the two byte combination DLE-DLE represents only a single data byte of 10H.

Flow control is accomplished using some hardware features of the TWC and the serial inter face on the microcomputer. The RTS (Request to Send) and CTS (Clear to Send) lines are cross connected and controlled by the programs. 'When the output line is high it means 'You can send data to me now'. When the output line is low it means 'Don't send any data to me now.'

Error detection is accomplished using the two-byte CRC (Cyclic Redundancy Check) characters following the FTX character in the block. I am using the following polynomial to generate the CRC bytes:

$$x^{16} + x^{15} + x^{2} + 1$$

This is the usual polynomial used for synchronous protocols such as IBM BISYNC but is not the one suggested by the CCITT. On transmit, the CRC calculation is done on all transmitted characters after the STX and up to and including the ETX character. The stuffed bytes are included in the calculation and after the STX is processed, two bytes of zeroes are processed. On receive, the calculation is the same except that the two CRC bytes are used instead of the zero bytes and the result of the CRC calculation will then come out to zero if everything was received correctly.

The error correction mechanism employed also utilizes some of the hardware features of the TNC and the microcomputer. The DTR (Data Terminal Ready) and the DSR (DataSet Ready) lines are cross connected between the TNC and microcomputer. Whenever one side receives a block correctly, it reverses the state of its output line. If the other side does not detect the transition then, after a timeout, it retransmits the block.

Hardware Requirements

In order to use the program called 'TIPTTC' which runs in the VADCG TNC, you will need a VADCG TNC with the serial interface installed and an RS232 cable with wires going to the following pins installed (2,3,4,5,6,7 and 20).

In order to use the program called 'PACKET' which runs in a CP/M system, you will need to have a serial interface capable of handling 8-bit characters, direct software control of two lines of RS-232 levels, and the ability to read two input RS-232 lines with the software. Most CP/M systems have this capability. It is true that I could have written this software to only require the data lines (and I may yet do this) and the software would be slightly more transportable but more complicated and a little less efficient. The flow control and acknowledgment systems work very well because the software in the TNC is alerted by the interrupt system almost instantly when there is any change in level of the interface lines.

Software Requirements

The 'TIPTTC' program should interface with any of the common LIP programs being used with the VADCG board. I can only think of one thing to watch out for - the program uses variables in the CCA (Common Communications Area) from displacement 40H to 54H so you should check your LIP's usage of these areas and relocate them if your LIP uses part of the same area. Also, make sure your stack does not get extended down as low as displacement 54H in the CCA. This is a 'vanilla' TIP and in addition to the features described above, it only has provision for connect and disconnect. If you use this TIP you will have to do without those special functions you previously had. The other alternative is to add the functions to this program yourself. If you take this latter option I would very much like to hear from you as well as anyone else who uses these programs. I like to get 'feedback.'

The 'PACKET' program only needs a CP/M system with the aforementioned hardware features and some configuration modifications described in the next section.

Confiauration Requirements

A. TIPTTC

A.1 At label 'BAUDRAT' the Baud rate may have to be changed. I am using 4800 Baud. In general it is best to have the rate as high as is reliable and convenient and should be 1200 or greater. However, lower Baud rates than 1200 would work as well.

A.2 At label 'ACKTO' there is a number which is related to the amount of time the TNC waits before retransmitting the block if no acknowledgment is received. This value has not been optimized from the first trial value. It is very non-critical and the value I chose for my system seems to work very well. It is probably quite a bit slower than required. You may experiment with different values.

A.3 At label 'RIMBUF' change the call sign to your own and if it is less than 6 characters, pad it on the right with blanks. Also, use upper case characters.

A.4 At label 'TERMNO' change your node number to whatever you want.

B. PACKET

B.1 In the section headed 'HARDWARE PORT EQUATES' you will have to change the port addresses to match the ports on your system.

B.2 In the sections headed 'CONTROL PORT BIT MEANINGS' and 'STATUS BIT MEANINGS' you will have to change the equates to match your system.

B.3 At label 'UARTINIT' change the code to initialize your serial interface UART to operate with 8 data bits and no parity bit. Also make the output lines going to pins 4 and 20 on the TNC are low. (The assumption here is that the jumper plug on the TNC is wired straight across)

B.4 At label 'SETRTS' change the code so that it makes pin 4 on the TNC end of the cable high.

B.5 At label 'CLEARRTS' change the code so that it makes pin 4 on the TNC end of the cable low.

B.6 At label 'FLIPDTR' make sure the code reverses the level on pin 20 of the TNC.

B.7 At label 'TESTTBE' test if data can be sent out to the UART and return non-zero status if it can.

B.8 At label 'TESTRDA' test if data is available from the UART and return non-zero status if it is.

B.9 At label 'TESTCTS' test the level of pin 5 coming from the TNC and return non-zero status if it is high.

B.10 At label 'TESTDSR' test the level of pin 6 coming from the TNC and compare it to the last tested level. If the value has changed, return non-zero status.

B.11 In routine 'KEYTEST' change the code to Look for a character to be entered on your keyboard and if there is none, then go to 'OUTTEST'. It will probably have to be changed because my keyboard uses inverted logic.

Operation

Although the importance of the "PACKET' program lies in the features provided by the drivers in it, I have added 25 instructions which allow the program to provide an immediately useful function. It will allow the user to use the keyboard and screen display in the CP/M system as if it were a terminal connected directly to the TNC. Because of the power of the driver code, it is a relatively trivial matter to add this function. Similarrily, a program to transfer a file from the system or to the system is very easy to implement using the drivers.

To use the program as a terminal simulator, simply type in a line of data on the keyboard, the line will be sent in a block to the TNC when the line feed key is pressed. While data is being entered after the first character, no blocks will be received from the TNC. While a block is being received from the TNC, the keyboard is not tested so a line that you enter will not be mixed with data coming from the TNC.

To connect, type the Call sign in upper case (which must be padded with blanks on the right if it is not 6 characters long) followed by control-A and then hit line feed to send it to the TNC. To disconnect, type any 6 characters (except for line feed) followed by control-B and then hit line feed. Sorry for this kludge but it is only temporary as I am planning to completely change the connect-disconnect procedures when I write the Transmission Control Program which is the next step in implementation of the Packet level protocols.

Summary

I hope these programs help those who are working on the implementation of the higher level protocols for an Amateur Radio digital communications network. It seemed to me that a program with these features would have to be one of the first steps in any kind of implementation but so far I have not heard of one. Perhaps someone out there has already written one and I have duplicated his effort. If so, then we are not doing enough advertising about what we have done. That is why I have taken this effort to disseminate the program.

The program listings here represent programs that have actually been running successfully so any problems encountered in transporting them to another system should be associated with the diffferent environment and not with defects in the code. I can supply the programs on standard SS-SD CP/M format diskettes if necessary. Please enclose $3.00 with a blank diskette or $8.00 without a diskette when making your request. You will find the listings for the two programs on the following pages.

* CP/M is a trade mark of Digital Research

```
                  *******************************************************
                  **      VADCG PACKET LEVEL TNC DRIVER FOR CP/M       **
                  **   BY DOUG LOCKHART, VE7APU         JANUARY, 1983  **
                  *******************************************************
                  ; LAST CHANGED JANUARY 31, 1983

                  *******************************************************
                  *  THIS PROGRAM CONTAINS THE DRIVERS TO EXCHANGE TRANS- *
                  *  PARENT BLOCKS OF DATA BETWEEN A CP/M OPERATING       *
                  *  SYSTEM AND A VADCG TERMINAL NODE CONTROLLER USING A *
                  *  MATCHING  PROGRAM.  IT USES THE REQUEST TO SEND (RTS) *
                  *  AND CLEAR TO SEND (CTS) LINES FOR FLOW CONTROL AND  *
                  *  THE DATA SET READY (DSR) AND DATA TERMINAL READY    *
                  * (DTR) LINES FOR ACKNOWLEDGEMENTS.  ONLY DATA INFOR-  *
                  *  MATION IS PASSED ON THE DATA LINES.  THE PROGRAM    *
                  *  USES 'BYTE STUFFING' TO ACHIEVE DATA TRANSPARENCY   *
                  *  AND USES A CRC-16 TO DETECT ERRORS.  IF THE TRANS-  *
                  *  MITTED DATA IS NOT ACKNOWLEDGED BY A CHANGE IN LEVEL *
                  *  THEN THE BLOCK IS SENT AGAIN.                        *
                  *******************************************************
                          MISCELLANEOUS EQUATES
0005 =            Los     EQU     5
                  ;
                  ;       ASCII EQUATES
000D =            CR      EQU     0DH     ; CARRIAGE RETURN
000A =            LF      EQU     0AH     ; LINE FEED
0010 =            DLE     EQU     10H     ; DATA LINK ESCAPE
0002 =            STX     EQU     02H     ; START OF TEXT
0003 =            ETX     EQU     03H     ; END OF TEXT
0016 =            SYN     EQU     16H     ; SYNCHRONIZING CHARACTER
00FF =            PAD     EQU     0FFH    ; PAD CHARACTER
                  *******************************************************
                  ;       HARDWARE PORT EQUATES
0001 =            DATA    EQU     01H             ; UART DATA PORT
0000 =            CONTROL EQU     00H             ; UART CONTROL PORT
0000 =            STATUS  EQU     00H             ; UART STATUS PORT
0002 =            KEYBD   EQU     02H             ; KEYBOARD DATA PORT
                  ;
                  ;       CONTROL PORT BIT MEANINGS
0001 =            DTR     EQU     01H             ; NOT DATA TERMINAL READY
0002 =            RTS     EQU     02H             ; NOT REQUEST TO SEND
0004 =            BRS0    EQU     04H             ; BAUD RATE SELECT
0008 =            BRS1    EQU     08H             ; BAUD RATE SELECT
0010 =            WLS1    EQU     10H             ; WORD LENGTH SELECT
0020 =            WLS2    EQU     20H             ; WORD LENGTH SELECT
0040 =            SBS     EQU     40H             ; STOP BIT SELECT
0080 =            PI      EQU     80H             ; PARITY INHIBIT
                  ;
                  ;       STATUS BIT MEANING
0001 =            RDA     EQU     01H             ; RECEIVE DATA AVAILABLE
0002 =            KSTB    EQU     02H             ; NOT KEYBOARD STROBE
0004 =            PE      EQU     04H             ; PARITY ERROR
0008 =            FE      EQU     08H             ; FRAMING ERROR
0010 =            OE      EQU     10H             ; OVERRUN ERROR
0020 =            DSR     EQU     20H             ; NOT DATA SET READY
0040 =            CTS     EQU     40H             ; NOT CLEAR TO SEND
0080 =            TBE     EQU     80H             ; TRANSMIT BUFFER EMPTY

                  *******************************************************
0100                      ORG     100H
0100 31C903               LXX     SP,STACK        ; INITIALIZE STACK
0103 CD4301               CALL    UARTINIT        ; INITIALIZE UART
0106 CD3501       OUTTEST:CALL    WRITESTAT       ; ANY DATA IN TBUF?
0109 CA2301               JZ      LINETEST        ; NO, TRY TO RECEIVE SOME
010C DB00         KEYTEST:IN      STATUS          ; ANY KEYBOARD DATA?
010E E602                 AN1     KSTB
0110 C20601               JNZ     OUTTEST         ; NO, TEST FOR LINE DATA
0113 DB02                 IN      KEYBD           ; GET DATA
0115 CD3A01               CALL    DISPLAY         ; DISPLAY IT
0118 CD3A05               CALL    WRITE           ; PUT IT INTO BUFFER
011B FEOA                 CPI     LF              ; WAS IT A LINE FEED?
011D CC1005               CZ      TCLOSE          ; YES,SEND DATA IN BUFFER
```

```
0120 C30601               JMP     OUTTEST         ; GO FOR MORE DATA
                  LINETEST:
0123 CD3505               CALL    READSTAT        ; DATA IN RECEIVE BUFFER?
0126 CC3A04               CZ      BLOCKRX         ; NO, TRY TO RECEIVE SOME
0129 CA0C01               JZ      KEYTEST         ; NO, TEST KEYBOARD ENTRY
012C CD1805               CALL    READ            ; GET DATA BYTE FROM RBUF
012F CD3A01               CALL    DISPLAY         ; AND DISPLAY IT
0132 C32301               JMP     LINETEST
                  WRITESTAT:
0135 3A9B02               LDA     TBUFNUM         ; GET COUNT
0138 B7                   ORA     A               ; AND TEST IT
0139 C9                   RET

013A F5           DISPLAY:PUSH    PSW
013B 5F                   MOV     E,A
013C OE02                 MVI     C,2
013E CD0500               CALL    BIOS            ; DISPLAY DATA IN (E)
0141 F1                   POP     PSW
0142 C9                   RET                     ; RETURN TO CALLER
                  *******************************************************
                  ;       BASIC UART DRIVER ROUTINES
                  ;
                  ;       INITIALZATION OF UART
                  UARTINIT:
0143 3EB7                 MVI     A,PI+WLS1+WLS2+BRS0+DTR+RTS     ; 8 DATA,
0145 D300                 OUT     CONTROL ; NO PARITY, DTR AND RTS OFF
0147 329801               STA     CTRL            ; SAVE CONTROL INFO
014A DB01                 IN      DATA            ; CLEAR ANY RESIDUAL DATA
014C DB00                 IN      STATUS          ; SAVE INITIAL DSR STATUS
014E E620                 ANI     DSR
0150 329701               STA     DSRSTAT
0153 C9                   RET                     ; RETURN TO CALLER

                  ;       ENABLE RTS (MEANS DATA CAN BE RECEIVED)
0154 3A9801       SETRTS: LDA     CTRL            ; GET CONTROL INFORMATION
0157 E6FD                 AN1     OFFH-RTS
0159 D300                 OUT     CONTROL
015B 329801               STA     CTRL
015E C9                   RET

                  ;       DISABLE RTS ( MEANS DO NOT SEND ME ANY DATA )
                  CLEARRTS:
015F 3A9801               LDA     CTRL            ; GET CONTROL INFORMATION
0162 F602                 ORI     RTS
0164 8300                 OUT     CONTROL
0166 329801               STA     CTRL
0169 C9                   RET

                  ;       REVERSE VALUE OF DTR (TO ACKNOWLEDGE BLOCK )
016A 3A9801       FLIPDTR:LDA     CTRL            ; GET CONTROL INFORMATION
016D EE01                 XRI     DTR             ; FLIP DTR
016F D300                 OUT     CONTROL
0171 329801               STA     CTRL    ; SAVE UART CONTROL INFORMATION
0174 C9                   RET                     ; RETURN TO CALLER

                  ;       TEST VALUE OF TBE (TRANSMIT BUFFER EMPTY)
0175 DB00         TESTTBE:IN      STATUS
0177 E680                 ANI     TBE
0179 C9                   RET

                  ;       TEST IF RECIEVE DATA IS AVAILABLE
017A DB00         TESTRDA:IN      STATUS
017C E601                 ANI     RDA
017E C9                   RET

                  ;       TEST VALUE OF CLEAR TO SEND
                  ;       NON-ZERO FLAG IF CTS, ZERO FLAG IF NO CTS
017F DB00         TESTCTS:IN      STATUS
0181 E640                 AN1     CTS
0183 FE40                 CPI     CTS             ; NOTE SENSE INVERTED
0185 C9                   RET
```

```
                ;       TEST IF VALUE OF DATA SET READY HAS CHANGED
                ;       NON-ZERO FLAG IF DSR HAS CHANGED, ZERO IF NOT
0186 E5         TESTDSR:PUSH    H               ; DO NOT CHANGE HL
0187 C5                 PUSH    B               ; OR BC
0188 219701             LX1     H,DSRSTAT       ; POINT AT OLD DSR STATUS
018B 46                 MOV     B,M
018C DB00               IN      STATUS
018E E620               AN1     DSR
0190 77                 MOV     M,A             ; SAVE NEW DSR STATUS
0191 B8                 CMP     B               ; COMPARE OLD AND NEW
0192 CI                 POP     B               ; RESTORE REGISTERS
0193 E1                 POP     H
0194 C9                 RET                     ; RETURN WITH FLAGS SET
                *********************************************************
0195 0000       CRC:    DW      0               ; CRC CALCULATION AREA
0197 00         DSRSTAT:DB      0               ; DSR STATUS SAVE AREA
0198 00         CTRL:   DB      0       ; CONTROL PORT INFORMATION
OOFA =          MAXNUM  EQU     250     ; MAXIMUM AMOUNT OF DATA ALLOWED
0199 9C01       RPOINT: DW      RBUF    ; NEXT POINT TO GET DATA IN RBUF
019B 00         RBUFNUM:DB      0       ; NUMBER OF BYTES IN RBUF
019C            RBUF:   DS      253             ; RECEIVE BUFFER
0299 9C02       TPOINT: DW      TBUF    ; NEXT POINT TO PUT DATA TN TBUF
029B 00         TBUFNUM:DB      0       ; NUMBER OF BYTES TN TBUF
029C            TBUF:   DS      253             ; TRANSMIT BUFFER
0399                    DS      30H             ; STACK AREA
03C9 =          STACK   EQU     $
                *********************************************************
                ;       SEND BYTE OF DATA OUT TO SERIAL PORT
                ;       DATA PASSED IN ACCUMULATOR

                SENDDATA:
03C9 CD5205             CALL    CALCCRC         ; INCLUDE IN CRC
03CC E5         SEND:   PUSH    H
03CD C5                 PUSH    B               ; SAVE B&C
03CE 4F         SEND1:  MOV     C,A             ; SAVE DATA TEMPORARILY
03CF 210100             LX1     H,1             ; DELAY FOR BUG IN UART
03D2 2B         SEND2:  DCX     H
03D3 7C                 MOV     A,H
03D4 B5                 ORA     L               ; IS IT o?
03D5 C2D203             JNZ     SEND2
03D8 CD7501     SEND3:  CALL    TESTTBE         ; IS TBUF EMPTY?
03DB CAD803             JZ      SEND3   ; NO, KEEP LOOPING UNTIL IT TS.
03DE CD7F01     SEND4:  CALL    TESTCTS         ; IS CLEAR TO SEND UP?
03E1 CADE03             JZ      SEND4           ; NO, CAN'T SEND YET
03E4 79                 MOV     A,C             ; GET BACK DATA BYTE
03E5 D301               OUT     DATA
03E7 C1                 POP     B               ; RESTORE B
03E8 E1                 POP     H
03E9 C9                 RET                     ; RETURN TO CALLER

                ;       SEND DATA IN TBUF TO THE UART TRANSPARENTLY

                SENDTBUF:
03EA E5                 PUSH    H
03EB C5                 PUSH    B
03EC 219B02             LX1     H,TBUFNUM       ; POINT TO TBUF BYTE CNT
03EF 4E                 MOV     C,M             ; SAVE IN C
                SENDTBUF1:
03F0 23                 INX     H               ; POINT TO NEXT BYTE
03F1 7E                 MOV     A, M            ; GET IT
03F2 CDC903             CALL    SENDDATA        ; SEND IT
03F5 FE10               CPI     DLE             ; WAS IT DLE?
03F7 CCC903             CZ      SENDDATA        ; IF SO, SEND IT AGAIN
03FA 0D                 DCR     C               ; DECREMENT COUNT
03FB C2F003             JNZ     SENDTBUF?       ; CONTINUE SENDING
03FE C1                 POP     B
03FF El                 POP     H
0400 C9                 RET                     ; RETURN TO CALLER

                  SEND FORMATTED BLOCK TO UART
                BLOCKTX:
0401 3E16               MVI     A,SYN
```

```
0403 CDCC03             CALL    SEND
0406 3E10               MVI     A,DLE
0408 CDCC03             CALL    SEND
040B 3E02               MVI     A,STX
040D CDCC03             CALL    SEND
0410 210000             LX1     H,0             ; INITIALIZE CRC AREA
0413 229501             SHLD    CRC
0416 CDEA03             CALL    SENDTBUF        ; SEND DATA IN TBUF
0419 3E10               MVI     A,DLE           ; THEN DLE-ETC SEQUENCE
041B CDC903             CALL    SENDDATA        ; INCLUDE IN CRC
041E 3E03               MVI     A,ETX
0420 CDC903             CALL    SENDDATA        ; INCLUDE IN CRC
0423 CDDE04             CALL    SENDCRC         ; FINALLY SEND CRC BYTES
0426 CDF204             CALL    CHECKRX         ; TRY TO RECEIVE
0429 CDD004             CALL    WAITDSR         ; WAIT FOR DSR TO CHANGE
042C CA0104             JZ      BLOCKTX ; DIDN'T CHANGE, SEND BLOCK AGAIN
042F AF                 XRA     A               ; A <-- 0
0430 329B02             STA     TBUFNUM         ; INDICATE TBUF IS EMPTY
0433 219C02             LX1     H,TBUF          ; POINT TO START OF TBUF
0436 229902             SHLD    TPOINT
0439 C9                 RET                     ; RETURN TO CALLER

                ;       READ A FORMATTED TRANSPARENT BLOCK OF DATA
043A CD5401     BLOCKRX:CALL    SETRTS          ; ALLOW OTHER END TO SEND
                BLOCKRX1:
043D CDAE04             CALL    RECEIVE         ; READ A BYTE FROM LINE
0440 C8                 RZ      ; RETURN WITH ZERO STATUS IF TIMED OUT
0441 FE10               CPI     DLE             ; IS IT DLE?
0443 C23D04             JNZ     BLOCKRX1        ; NO, KEEP TRYING
0446 CDAE04             CALL    RECEIVE         ; GOT DLE, TRY FOR STX
0449 C8                 RZ       ; RETURN WITH ZERO STATUS IF TIMED OUT
044A FE02               CPI     STX             ; IS IT STX?
044C C23D04             JNZ     BLOCKRX1        ; NO, TRY FOR DLE AGAIN
                BLOCKRX2:                       ; ENTRY FROM BLOCKTX
044F 219C01             LX1     H,RBUF          ; POINT TO START OF RBUF
0452 229901             SHLD    RPOINT
0455 CD8004             CALL    RCVRBUF         ; RECEIVE DATA INTO RBUF
                                ; UNTIL A CONTROL SEQUENCE IS RECEIVED
0458 C8                 RZ      ; RETURN ZERO STATUS IF LINE TIMES OUT
0459 FE03               CPI     ETX             ; WAS IT ETX?
045B C23D04             JNZ     BLOCKRX1        ; UNEXPECTED SEQUENCE
045E CDAE04             CALL    RECEIVE         ; RECEIVE FIRST CRC CHAR
0461 C8                 RZ                      ; RETURN HERE IF TIME OUT
0462 CDAE04             CALL    RECEIVE         ; RECEIVE SECOND CRC CHAR
0465 C8                 RZ                      ; RETURN HERE IF TIME OUT
0466 CD5F01             CALL    CLEARRTS        ; STOP OTHER END
0469 2A9501             LHLD    CRC             ; CHECK IF CRC WAS OK
046C 7C                 MOV     A,H
046D B5                 ORA     L
046E C27C04             JNZ     BLOCKRX3        ; NO GOOD
0471 219B01             LX1     H,RBUFNUM       ; SAVE DATA COUNT
0474 71                 MOV     M,C
0475 CD6A01             CALL    FLIPDTR         ; GOOD, REVERSE DTR LINE
                                                ; TO ACKNOWLEDGE BLOCK
0478 3EFF               MVI     A,-1            ; RETURN NON-ZERO STATUS
047A B7                 ORA     A               ; BLOCK RECEIVED OK
047B C9                 RET                     ; RETURN TO CALLER
                BLOCKRX3:
047C 3E00               MVI     A,0             ; RETURN WITH ZERO STATUS
047E B7                 ORA     A               ; NO BLOCK RECEIVED
047F C9                 RET


                i       RECEIVE DATA PORTION OF BLOCK, RETTJRNS WHEN A
                ;       CONTROL SEQUENCE FOUND IN THE TRANSPARENT TEXT
0480 210000     RCVRBUF:LXI     H,0             ; INITIALIZE CRC TO 0
0483 229501             SHLD    CRC
0486 219C01             LXI     H,RBUF          ; POINT TO START OF RBUF
0489 0E00               MVI     C,0             ; BYTE COUNT = 0
                RCVRBUF1:
048B CDAE04             CALL    RECEIVE         ; GET A BYTE FROM LINE
048E C8                 Rz      ; RETURN HERE WITH ZERO STATUS IF TIMEOUT
```

```
048F FE10               CPI     DLE             ; WAS IT DLE?
0491 CA9A04             JZ      RCVRBUF3        ; YES, LOOK AT NEXT BYTE
              RCVRBUF2:
0494 77                 MOV     M,A             ; PUT INTO BUFFER
0495 23                 INX     H               ; INCREMENT RBUF POINTER
0496 OC                 INR     C               ; INCREMENT COUNT
0497 C38B04             JMP     RCVRBUF1        ; LOOP FOR NEXT BYTE
              RCVRBUF3:
049A CDAE04             CALL    RECEIVE
049D C8                 RZ      ; ZERO STATUS RETURN IF LINE TIMES OUT
049E FE10               CPI     DLE     ; IS IT A TRANSPARENT DLE?
04A0  CA9404            JZ      RCVRBUF2        ; YES, GO PUT INTO BUFFER
04A3 C9                 RET     ; RETURN WITH CONTROL BYTE IN ACCUMULATOR
                                                ; AND NON-ZERO STATUS

              ;         TRY TO READ PROM LINE WITH LONG TIMEOUT
04A4 E5       RECEIVL:PUSH      H
04A5 CD5401             CALL    SETRTS          ; ALLOW OTHER END TO SEND
04A8  21A00F            LXI     H,4000          ; LONG TIMEOUT VALUE
                                ; SHOULD BE ADJUSTED FOR BEST RESULTS
04AB C3B504             JMP     RECEIV1

              ;         TRY TO READ FROM LINE, IF LINE TIMES OUT,
              ;         RETURN WITH ZERO STATUS
04AE E5       RECEIVE:PUSH      H
04AF CD5401             CALL    SETRTS          ; ALLOW OTHER END TO SEND
04B2 210007             LX1     H,2000          ; SHORT TIMEOUT VALUE,
                                ; ADJUST FOR ABOUT 2 CHAR TIMES
04B5 CD7A01   RECEIV1:CALL      TESTRDA         ; ANY RECEIVED DATA?
0488 CAC204             JZ      RECEIV2         ; NO, DECREMENT TIME
04BB DBO?               IN      DATA            ; GET DATA BYTE
04BD CD5205             CALL    CALCCRC         ; INCLUDE IT IN CRC
04C0 El                 POP     H
04C1 C9                 RET     ; GOOD RETURN WITH NON ZERO STATUS
04C2 2B       RECEIV2:DCX       H               ; DECREMENT TIMER
04C3 7C                 MOV     A,H
04C4 B5                 ORA     L               ; IS TIME OVER
04C5 C2B504             JNZ     RECEIV1         ; NO, KEEP TRYING
04C8 CD5F01             CALL    CLEARRTS        ; OOPS, TIMED  OUT,
                                ; DROP RTS SO OTHER SIDE WILL STOP
04CB 3E00               MVI     A,0     ; RETURN WITH ZERO STATUS
04CD B7                 ORA     A
04CE E1                 POP     H
04CF C9                 RET

              ;         WAIT FOR DSR TO CHANGE USING TIMEOUT
              WAITDSR:
04D0 011027             LXI     B, 10000        ; DELAY - ALTER AS REQ'D
              WAITDSR?:
04D3 CD8601             CALL    TESTDSR         ; CHECK FOR DSR CHANGE
04D6 CO                 RNZ                     ; RETURN IF IT HAS
04D7 0B                 DCX     B
04D8 78                 MOV     A,B
04D9 B1                 ORA     C               ; IS TIME OVER?
04DA C2D304             JNZ     WAITDSR1        ; NO CONTINUE TESTING
04DD C9                 RET                     ; UNSUCCESSFUL RETURN

              ;         SENDTHECRC  BYTES
04DE AF       SENDCRC:XRA       A               ; FINISH CRC CALCULATION
04DF CD5205             CALL    CALCCRC
04E2 CD5205             CALL    CALCCRC
04E5 3A9601             LDA     CRC+1           ; SEND FIRST CRC CHAR
04E8 CDCC03             CALL    SEND
04EB 3A9501             LDA     CRC             ; SEND SECOND CRC CHAR
04EE  CDCC03            CALL    SEND
04F1 C9                 REX

              ;         CHECK IF WE CAN RECEIVE A BLOCK NOW
              ;         THIS ROUTINE IS CALLED AFTER A BLOCK HAS BEEN
              ;         TRANSMITTED TOALLOW THE OTHER SIDE TO GET A
              ;         CHANCE TO SEND TO US
04F2 3A9B01   CHECKRX: LDA      RBUFNUM         ; IS THERE ANY DATA LEFT
```

```
04F5 B7                 ORA     A               ; IN RECEIVE BUFFER?
04F6 C0                 RNZ                     ; YES, CAN'T RECEIVE
04F7 CD5401             CALL    SETRTS          ; ENABLE RTS TO ALLOW
                                                ; OTHER SIDE TO SEND
              CHECKRX1:
04FA CDA404             CALL    RECEIVL         ; READ WITH LONG TIMEOUT
04FD  C8                RZ                      ; TIMED OUT , RETURN
04FE  FE10              CPI     DLE             ; IS IT A DLE?
0500 C2FA04             JNZ     CHECKRX1        ; NO, KEEP LOOKING
0503  CDA404            CALL    RECEIVL         ; NOW LOOK FOR A STX
0506  C8                RZ                      ; TIMED OUT SO RETURN
0507  FE02              CPI     STX             ; IS IT START OF TEXT?
0509 C2FA04             JNZ     CHECKRX1        ; NO, KEEP LOOKING
050C CD4F04             CALL    BLOCKRX2        ; NOW GO READ TRANSP.TEXT
050F C9                 RET                     ; ZERO STATUS IF TIMEOUT
                                        ; NON-ZERO IF BLOCK WAS RECEIVED

              ;         SEND A BLOCK OF TRANSMIT DATA TO THE LINE I?
              ;         THERE IS ANY DATA IN THE BUFFER
0510 3A9B02   TCLOSE: LDA       TBUFNUM         ; GET COUNT IN BUFFER
0513 B7                 ORA     A               ; TEST FOR DATA
0514 C40104             CNZ     BLOCKTX         ; SEND BLOCK IF ANY DATA
0517 C9                 RET                     ; RETURN TO CALLER

0518 ES       READ:     PUSH    H               ; SAVE HL
0519 219B01   READ1:    LXI     H,RBUFNUM       ; POINT AT COUNT IN RBUF
051C 7E                 MOV     A,M
051D B7                 ORA     A               ; IS THERE MY LEFT?
051E C22A05             JNZ     READ2           ; YES
0521 CD1005             CALL    TCLOSE          ; SEND ANY DATA IN TBUF
0524 CD3A04             CALL    BLOCKRX         ; RECEIVE ANOTHER BLOCK
0527 C31905             JMP     READ1           ; TRY TO DO READ AGAIN
052A  35      READ2:    DCR     M               ; DECREMENT COUNT
052B 2A9901             LHLD    RPOINT          ; GET READ POINTER
052E  7E                MOV     A,M             ; GET DATA BYTE
052F  23                INX     H               ; INCREMENT POINTER
0530   229901           SHLD    RPOINT          ; AND SAVE AGAIN
0533 El                 POP     H               ; RESTORE HL
0534 C9                 RET             ; RETURN TO CALLER WITH DATA IN A

              READSTAT:
0535 3A9B01             LDA     RBUFNUM ; GET COUNT OF DATA IN BUFFER
0538 B7                 ORA     A               ; TEST IT
0539 C9                 RET             ; NON-ZERO STATUS IF DATA PRESENT

053A F5       WRITE:    PUSH    PSW             ; SAVE DATA
053B E5                 PUSH    H               ; SAVE HL
053C 2A9902             LHLD    TPOINT          ; GET POINTER INTO TBUF
053F  77                MOV     M,A             ; PUT DATA INTO BUFFER
0540  23                INX     H               ; INCREMENT POINTER
0541   229902           SHLD    TPOINT
0544 219B02             LX1     H,TBUFNUM       ; POINT To COUNT IN TBUF
0547  7E                MOV     A,M             ; INCREMENT COUNT
0548  3C                INR     A
0549  77                MOV     M,A
054A  FEFA              CPI     MAXNUM          ; IS BUFFER FULL?
054C cc0104             CZ      BLOCKTX         ; YES, SEND BLOCK NOW
054F E?                 POP     H               ; RESTORE HL
0550 F1                 POP     PSW             ; RESTORE  DATA
0551  C9                RET

              ; CRC CALCULATION ROUTINE
              ; USES BYTE PASSED IN ACCUMULATOR TO INCLUDE IN CRC
              ; RESTORES ALL REGISTERS AND STATUS
0552 E5       CALCCRC:PUSH      H
0553  c5                PUSH    B
0554  F5                PUSH    PSW
0555  0608              MVI     B,8
0557  4F                MOV     C,A
0558 2A9501             LHLD    CRC
055B =        CALCCRC1:EQU      $
055B 79                 MOV     A,C
```

```
055C 07                 RLC
055D 4F                 MOV     C,A
055E 7D                 MOV     A,L
055F 17                 RAL
0560 6F                 MOV     L,A
0561 7C                 MOV     A,H
0562 17                 RAL
0563 67                 MOV     H,A
0564 D26F05             JNC     CALCCRC2
0567 7C                 MOV     A,H
0568 EE80               XRI     80H
056A 67                 MOV     H,A
056B 7D                 MOV     A,L
056C EE05               XRI     05H
056E 6F                 MOV     L,A
               CALCCRC2:
056F 05                 DCR     B
0570 C25B05             JNZ     CALCCRC1
0573 229501             SHLD    CRC
0576 F1                 POP     PSW
0577 CI                 POP     B
0578 El                 POP     H
0579 C9                 RET

057A                    END
```

```
                ; ************************************************************
                ; **    VADCG TERMINAL NODE COMMUNICATIONS PROGRAM          **
                ; **    BY DOUG LOCKHART, VE7APU     JANUARY, 1983          **
                ; ************************************************************
                ; LAST CHANGED: JANUARY 31, 1983

                ; TERMINAL INTERFACE PROGRAM FOR INTERFACING TO A CP/M
                ; SYSTEM.  THIS PROGRAM IS WRITTEN TO RUN IN THE VADCG
                ; TERMINAL NODE CONTROLLER.  IT INTERFACES WITH A LINK
                ; INTERFACE PROGRAM (LIP) RUNNING AT ADDRESS 0 IN MEMORY.
                ; THIS VERSION IS WRITTEN TO USE THE 8250 PROGRAMMABLE
                ; UART TO COMMUNICATE WITH A COMPUTER.
                ; THE BASIC FEATURES OF THIS TIP ARE:
                ; TRANSFER OF DATA IN BLOCKS
                ; RTS FLOW CONTROL FROM DIGITAL EQUIPMENT TO TIP
                ; AND CTS FLOW CONTROL FROM TIP TO DIGITAL EQUIPMENT
                ; ACKNOWLEDGEMENTS TO BLOCKS RECEIVED BY A CHANGE IN DTR
                ; ACKNOWLEDGEMENTS TO BLOCKS SENT BY A CHANGE IN DSR
                ; CRC-16 CHECKING OF ALL DATA BLOCKS
                ; ERROR RECOVERY BY RETRANSMISSION IF NO ACKNOWLEDGMENT
                ; USES BYTE STUFFING TECHNIQUE FOR DATA TRANSPARENCY

                INCTB   MACRO   ?D
                        IF      NOT NUL ?D
                        MVI     A,?D
                        ENDIF
                        RST     2
                        ENDM

                INCLB   MACRO   ?D
                        IF      NOT NUL ?D
                        MVI     A,?D
                        ENDIF
                        RST     3
                        ENDM

                COMPARE MACRO
                        RST     5
                        ENDM

                SIM     MACRO
                        DB      30H     ; SET INTERRUPT MASK
                        ENDM

                RIM     MACRO
                        DB      20H     ; READ INTERRUPT MASK
                        ENDM


                ; RAM CONSTANT - CHANGE FOR DIFFERENT RAM LOCATION
1000 =          LORAM   EQU     1000H   ; START OF RAM STORAGE


                ; NON-ZERO STATUS MEANS LINE BUFFER ADDRESS IS IN HL REG.
                ; ZERO STATUS MEANS NO BUFFER IS READY
                NEXTIN  MACRO
                        RST     4
                        ENDM


                ; 8255 PARALLEL I/O EQUATES

0008 =          PORTA   EQU     8       ; PORT A INPUT AND OUTPUT
0009 =          PORTB   EQU     9       ; PORT B INPUT AND OUTPUT
000A =          PORTC   EQU     0AH     ; PORT C INPUT AND OUTPUT
000B =          CONTROL EQU     0BH     ; CONTROL PORT OUTPUT ONLY


                ; BAUD RATE EQUATES
0004 =          BAUD384 EQU     4       ; DIVISOR FOR 38,400 BAUD
0008 =          BAUD192 EQU     8       ; DIVISOR FOR 19,200 BAUD
0010 =          BAUD96  EQU     16      ; DIVISOR FOR 9600 BAUD
```

```
0020 =          BAUD48  EQU     32      ; DIVISOR FOR 4800 BAUD
0040 =          BAUD24  EQU     64      ; DIVISOR FOR 2400 BAUD
0080 =          BAUD12  EQU     128     ; DIVISOR FOR 1200 BAUD
0100 =          BAUD600 EQU     256     ; DIVISOR FOR 600 BAUD
0200 =          BAUD300 EQU     512     ; DIVISOR FOR 300 BAUD
0400 =          BAUD150 EQU     1024    ; DIVISOR FOR 150 BAUD
0476 =          BAUD134 EQU     1142    ; DIVISOR FOR 134.5 BAUD
0573 =          BAUD110 EQU     1395    ; DIVISOR FOR 110 BAUD
0800 =          BAUD75  EQU     2048    ; DIVISOR FOR 75 BAUD
0C00 =          BAUD50  EQU     3072    ; DIVISOR FOR 50 BAUD


                ; 8250 SERIAL I/O EQUATES

                ; REGISTER EQUATES
0000 =          RBR     EQU     0       ; RECEIVE BUFFER REGISTER  (R)
0000 =          THR     EQU     0       ; TRANSMIT HOLDING REGISTER (W)
0001 =          IER     EQU     1       ; INTERRUPT ENABLE REGISTER  (W)
0002 =          IIR     EQU     2       ; INTERRUPT IDENT. REGISTER (R)
0003 =          LCR     EQU     3       ; LINE CONTROL REGISTER (R/W)
0004 =          MCR     EQU     4       ; MODEM CONTROL REGISTER (R/W)
0005 =          LSR     EQU     5       ; LINE STATUS REGISTER (R/W)
0006 =          MSR     EQU     6       ; MODEM STATUS REGISTER (R/W)
0000 =          DLL     EQU     0       ; DRIVER LATCH (LSB) (W)
0001 =          DLM     EQU     1       ; DRIVER LATCH (MSB) (W)

                ; INTERRUPT ENABLE EQUATES
0001 =          ERBFI   EQU     1       ; ENABLE RECEIVED DATA INTERRUPT
0002 =          ETBEI   EQU     2       ; ENABLE TRANSMITTER
0004 =          ELSI    EQU     4       ; RECEIVER LINE STATUS INTERRUPT
0008 =          EDSSI   EQU     8       ; ENABLE MODEM STATUS INTERRUPT

                ; INTERRUPT IDENTIFICATION EQUATES
0001 =          IPEND   EQU     1       ; '0' IF INTERRUPT PENDING
0002 =          IID0    EQU     2       ; INTERRUPT IDENTIFICATION BIT 0
0004 =          IID1    EQU     4       ; INTERRUPT IDENTIFICATION BIT 1

                ; LINE CONTROL EQUATES
0001 =          WLS0    EQU     1       ; WORD LENGTH SELECT BIT 0
0002 =          WLS1    EQU     2       ; WORD LENGTH SELECT BIT 1
0004 =          STB     EQU     4       ; STOP BIT SELECT
0008 =          PEN     EQU     8       ; PARITY ENABLE
0010 =          EPS     EQU     10H     ; EVEN PARITY SELECT
0020 =          SPTY    EQU     20H     ; STICK PARITY
0040 =          SBRK    EQU     40H     ; SET BREAK
0080 =          DLAB    EQU     80H     ; DRIVER LATCH ACCESS BIT

                ; MODEM CONTROL EQUATES
0001 =          DTR     EQU     1       ; DATA TERMINAL READY
0002 =          RTS     EQU     2       ; REQUEST TO SEND
0004 =          OUT1    EQU     4       ; OUT1 LINE ON 8250
0008 =          OUT2    EQU     8       ; OUT2 LINE ON 8250
0010 =          LOOP    EQU     10H     ; MODEM LOOP CONTROL BIT

                ; LINE STATUS EQUATES
0001 =          DR      EQU     1       ; DATA READY
0002 =          OE      EQU     2       ; OVERRUN ERROR
0004 =          PE      EQU     4       ; PARITY ERROR
0008 =          FE      EQU     8       ; FRAMING ERROR
0010 =          BI      EQU     10H     ; BREAK INTERRUPT
0020 =          THRE    EQU     20H     ; TRANSMITTER HOLDING REG EMPTY
0040 =          TSRE    EQU     40H     ; TRANSMITTER SHIFT REG EMPTY

                ; MODEM STATUS EQUATES
0001 =          DCTS    EQU     1       ; DELTA CLEAR TO SEND
0002 =          DDSR    EQU     2       ; DELTA DATA SET READY
0004 =          TERI    EQU     4       ; TRAILING EDGE RING INDICATOR
0008 =          DRLSD   EQU     8       ; DELTA RX LINE SIGNAL DETECT
0010 =          CTS     EQU     10H     ; CLEAR TO SEND
0020 =          DSR     EQU     20H     ; DATA SET READY
0040 =          RI      EQU     40H     ; RING INDICATE
```

```
0080 =          RLSD    EQU     80H      ; RECEIVE LINE SIGNAL DETECT

0017 =          RIMD    EQU     17H      ; REQUEST INITIALIZATION MODE
0008 =          MSE     EQU     08H      ; MASK SET ENABLE BIT


                ; COMMON COMMUNICATIONS AREA

                ; CIRCULAR TERMINAL BUFFER VARIABLES

1000 =          CCA     EQU     LORAM    ; COMMON COMMUNICATIONS AREA ADR.
1004 =          CTBIE   EQU     CCA+4    ; CURRENT TERMINAL BUF INP. ENTRY
1006 =          OTBE    EQU     CCA+6    ; OLDEST TERMINAL BUFFER ENTRY
1008 =          TBIP    EQU     CCA+8    ; TERMINAL BUFFER INPUT POINTER
100A =          TBOP    EQU     CCA+0AH  ; TERMINAL BUFFER OUTPUT POINTER
100C =          LTBOE   EQU     CCA+OCH  ; LAST TERMINAL BUF OUTPUT ENTRY
100E =          CTBOE   EQU     CCA+0EH  ; CURRENT TERMINAL BUF OUT ENTRY

                ; CIRCULAR LINE BUFFER VARIABLES

1012 =          LBPE    EQU     CCA+12H  ; LINE BUFFER PROCESSING ENTRY
1014 =          CLBE    EQU     CCA+14H  ; CURRENT LINE BUFFER ENTRY ADDR.
1016 =          OLBE    EQU     CCA+16H  ; OLDEST LINE BUFFER ENTRY
1018 =          LBIP    EQU     CCA+18H  ; LINE BUFFER INPUT POINTER
101A =          LBOP    EQU     CCA+1AH  ; LINE BUFFER OUTPUT POINTER

                ; MISCELLANEOUS


1000 =          STAT1   EQU     CCA      ; MAINLINE STATUS BYTE

                ; THE FOLLOWING VARIABLES ARE FOR EXCLUSIVE USE BY TIP

101C =          BUFCOUNT EQU    CCA+1CH  ; CURRENT INPUT BUFFER COUNT
101D =          OUTCOUNT EQU    CCA+1DH  ; CURRENT OUTPUT BYTES REMAINING
1040 =          WAIT    EQU     CCA+40H  ; CHARACTER DELAY VALUE
1042 =          MSRSAVE EQU     CCA+42H  ; LATEST MODEM STATUS REGISTER
1043 =          INTFLAG EQU     CCA+43H  ; INTERRUPT ROUTINE FLAGS
0001 =          RXBUSY  EQU     01H      ; RECEIVE INTRPT ROUTINE ACTIVE
0002 =          TXBUSY  EQU     02H      ; TRANSMIT INTRPT ROUTINE ACTIVE
1044 =          CRC     EQU     CCA+44H  ; CRC CALCULATION AREA
1046 =          RCRC2   EQU     CCA+46H  ; SECOND RECEIVED CRC BYTE
1047 =          RCRC1   EQU     CCA+47H  ; FIRST  RECEIVED CRC BYTE
1048 =          TCRC2   EQU     CCA+48H  ; SECOND TRANSMIT CRC BYTE
1049 =          TCRC1   EQU     CCA+49H  ; FIRST TRANSMIT CRC BYTE
104A =          RNEXT   EQU     CCA+4AH  ; CURRENT RECEIVE ROUTINE ADDRESS
104C =          TNEXT   EQU     CCA+4CH  ; TRANSMIT ROUTINE ADDRESS
104E =          RDISP   EQU     CCA+4EH  ; RECEIVE INTERRUPT ROUTINE ADDR.
1050 =          TDISP   EQU     CCA+50H  ; TRANSMIT INTERRUPT ROUTINE ADDR
1052 =          DFLAG   EQU     CCA+52H  ; DISPATCH FLAG
0001 =          CRCTX   EQU     01H      ; CRC ROUTINE IN USE BY TX DISP.

                ; ASCII EQUATES
000D =          CR      EQU     0DH      ; ASCII CARRIAGE RETURN
000A =          LF      EQU     0AH      ; ASCII LINE FEED
001B =          ESC     EQU     1BH      ; ASCII ESCAPE CHARACTER
0002 =          STX     EQU     02H      ; ASCII START OF TEXT
0003 =          ETX     EQU     03H      ; ASCII END OF TEXT
0010 =          DLE     EQU     10H      ; ASCII DATA LINK ESCAPE
0016 =          SYN     EQU     16H      ; ASCII SYNCHRONIZATION CHARACTER
00FF =          PAD     EQU     0FFH     ; TRAILING PAD CHARACTER

00FF =          TRUE    EQU     OFFH     ; FOR IF CONDITION TESTS
0000 =          FALSE   EQU     0        ; FOR IF CONDITION TESTS

                ; ***************************************************
                ; **            CONFIGURATION EQUATES              **
                ; **     VALUES CHANGE FOR EVERY CONFIGURATION     **
                ; ***************************************************


0003 =          FORMAT  EQU     WLS1+WLS0       ; UART FORMAT (8 DATA,
                                                 ; NO PARITY)
0020 =          BAUDRAT EQU     BAUD48   ; CURRENT BAUD RATE

00FF =          CUSHION EQU     255      ; THE MINIMUM NUMBER OF BYTES
                                ; AVAILABLE IN THE TERMINAL BUFFER THAT
                                ; ARE REQUIRED BEFORE A RECEIVE
                                ; OPERATION IS STARTED.

2710 =          ACKTO   EQU     10000    ; ACKNOWLEDGE TIMEOUT COUNT
                                         ; (PRELIMINARY VALUE)

                ***********************************************************
0800                    ORG     800H     ; THIS PROGRAMS EPROM START ADR.

                ; ENTRY JUMP TABLE

0800 C31508             JMP     TJPINIT ; INITIALIZATION ENTRY POINT
0803 C34808             JMP     RST55    ; INTERRUPT FROM 8250
0806 C30608             JMP     $        ; UNUSED INTERRUPT ENTRY POINT
0809 C3100A             JMP     DISPRX   ; TO DISPATCHER ROUTINE
080C 0C17564537RIMBUF   DB      12,RIMD,'VE7APU'        ; CONNECT BUFFER
0814 C8         TERMNO  DB      200      ; THIS NODES TERMINAL NUMBER
                ***********************************************************
                TIPINIT:
                        ,SET BAUD RATE IN SERIAL PORT
0815 3E80               MVI     A,DLAB
0817 D303               OUT     LCR
0819 3E20               MVI     A,LOW BAUDRAT
081B D300               OUT     DLL               ; BAUD RATE DIVISOR LSB
081D 3E00               MVI     A,HIGH BAUDRAT
081F D301               OUT     DLM               ; BAUD RATE DIVISOR MSB

                        ; DEFINE CHARACTER FORMAT OF SERIAL DATA
0821 3E03               MVI     A,FORMAT
0823 D303               OUT     LCR      ; UPDATE LINE CONTROL REGISTER

                        ; UNMASK INTERRUPTS FROM SERIAL INTERFACE
                        RIM              ; GET CURRENT INTERRUPT MASK IN A
0825+20                 DB      20H      ; READ INTERRUPT MASK
0826 E606               AN1     00000110B         ; RESET RST5.5 MASK BIT
0828 F608               OR1     MSE      ; SET MASK SET ENABLE BIT
                        SIM              ; ENABLE RST5.5 INTERRUPTS
082A+30                 DB      30H      ; SET INTERRUPT MASK

                        ; CLEAR OUT RECEIVE BUFFER REGISTER
082B DB00               IN      RBR

                        ; SET UP INITIAL DISPATCH ROUTINES
082D 214309             LX1     H,EXIT   ; SET RECEIVE INTERRUPT TO IDLE
0830 224A10             SHLD    RNEXT
0833 211608             LX1     H,WAITLIP         ; WAITING FOR LIP BLOCK
0836 225010             SHLD    TDISP
0839 2I140A             LXI     H'WAITTB          ; WAITING FOR FREE CUSHION
083C 224310             SHLD    RDISP
                ; ENABLE RECEIVED DATA AVAILABLE AND MODEM STATUS INTRPT
083F 3E09               MVI     A,ERBFI+EDSSI     ; RECEIVE AND MODEM
0841 D301               OUT     IER      ; UPDATE INTERRUPT REGISTER

                        ; BRING UP RLSD (OUT1 = RLSD)
0843 3E04               MVI     A,OUT1
0845 D304               OUT     MCR      ; UPDATE MODEM CONTROL REGISTER

                        ; RETURN TO LIP FOR COMPLETION OF INITIALIZATION
0847 C9                 RET
                ***********************************************************
0848 F5         RST55:  PUSH    PSW
0849 E5                 PUSH    H
084A D5                 PUSH    D
084B C5                 PUSH    B
084C DB02               IN      IIR      ; GET INTERRUPT IDENT INFORMATION
084E FE04               CPI     IID1     ; RECEIVED DATA AVAILABLE INTRPT?
0850 CA8A08             JZ      RXINT    ; GO TO RECEIVE INTERRUPT ROUTINE
```

```
0853 FE02               CPI     1100     ; IS IT TRANSMIT BUFFER EMPTY
0855 CA4909             JZ      TXINT    ; GO TO TRANSMIT INTRPT ROUTINE
0858 B7                 ORA     A        ; MODEM STATUS INTERRUPT?
0859 CA5F08             JZ      MSINT    ; TO MODEM STATUS INTRPT ROUTINE
085C C34309             JMP     EXIT     ; UNKNOWN INTERRUPT, RETURN

085F DB06       MSINT:  IN      MSR      ; GET MODEM STATUS
0861 324210             STA     MSRSAVE  ; SAVE MODEM STATUS FOR DISPATCH
0864 4F                 MOV     C,A      ; SAVE IT
0865 E601               AN1     DCTS     ; HAS CTS CHANGED?
0867 C46D08             CNZ     CTSINT   ; YES GO HANDLE CTS CHANGE
086A C34309             JMP     EXIT

086D 79         CTSINT: MOV     A,C      ; GET MODEM STATUS BACK
086E E610               AN1     CTS      ; TEST CTS BIT
0870 CA7608             JZ      DISABLETX         ; OFF, DISABLE TRANSMIT
0873 C37D08             JMP     ENABLETX          ; TRY TO ENABLE TRANSMIT

                DI SABLETX:
0876 DB01               IN      IER      ; GET INTERRUPT ENABLE REGISTER
0878 E6FD               AN1     0FFH-ETBEI
087A D301               OUT     IER      ; TURN OFF TRANSMIT INTERRUPTS
087C C9                 RET

                ENABLETX:
087D 3A4310             LDA     INTFLAG  ; IS TRANSMITTER BUSY?
0880 E602               ANI     TXBUSY
0882 C8                 RZ               ; NO, RETURN
0883 DB01               IN      IER      ; GET INTERRUPT ENABLE REGISTER
0885 F602               OR1     ETBEI
0887 D301               OUT     IER      ; ENABLE TRANSMIT INTERRUPTS
0889 C9                 RET
                ***********************************************************
088A DB00       RXINT:  IN      RBR      ; READ DATA FROM SERIAL PORT
088C 2A4A10             LHLD    RNEXT    ; GO TO ROUTINE ADDRESS IN RNEXT
088F E9                 PCHL

0890 FE10       RSTART: CPI     DLE      ; IS IT A DATA LINK ESCAPE?
0893 C24309             JNZ     EXIT     ; NO
0895 219E08             LX1     H,RSTX   ; YES, NOW WAIT FOR START OF TEXT
0898 224A10             SHLD    RNEXT
089B C34309             JMP     EXIT


089E FE02       RSTX:   CPI     STX      ; IS IT START OF TEXT
08A0 C2AC08             JNZ     RSTX1    ; NO
08A3 21B508             LX1     H,RDATA  ; YES, HANDLE TRANSPARENT DATA
08A6 224A10             SHLD    RNEXT
08A9 C34309             JMP     EXIT
08AC 219008     RSTX1:  LXI     H,RSTART          ; FALSE START GO BACK
08AF 224A10             SHLD    RNEXT             ; TO BEGINNING
08B2 C34309             JMP     EXIT

08B5 FE10       RDATA:  CPI     DLE      ; IS IT A DLE?
08B7 CAC308             JZ      RDATA1   ; YES
08BA CDEE08             CALL    RPUT     ; NO, PUT DATA INTO BUFFER
08BD CA0609             JZ      RESTART  ; ERROR, RESET BUFFER AND RESTART
08C0 C34309             JMP     EXIT              ; FROM BEGINNING
08C3 21CC08     RDATA1: LXI     H,RCONTROL        ; RECEIVE CONTROL
08C6 224A10             SHLD    RNEXT             ; CHARACTER NEXT
08C9 C34309             JMP     EXIT

                RCONTROL:
08CC FE10               CPI     DLE      ; IS IT A SECOND DLE?
08CE C2E008             JNZ     RCONTROL1         ; NO, CHECK FOR ETX
08D1 CDEE08             CALL    RPUT     ; YES, PUT DLE IN BUFFER
08D4 CA0609             JZ      RESTART  ; ERROR, RESET BUFFER AND RESTART
08D7 21B508             LX1     H,RDATA  ; GO BACK FOR MORE DATA
08DA 224A10             SHLD    RNEXT
08DD C34309             JMP     EXIT
                RCONTROL1:
08E0 FE03               CPI     ETX      ; IS IT END OF TEXT?
```

```
08E2 C20609               JNZ     RESTART ; NO, ERROR - RESTART
08E5 211D09               LXI     H,R1CRC ; NEXT RECEIVE FIRST CRC CHAR
08E8 224A10               SHLD    RNEXT
08EB C34309               JMP     EXIT

08EE 4F         RPUT:     MOV     C,A     ; SAVE DATA IN REGISTER C
08EF 2A0610               LHLD    OTBE    ; PUT DATA INTO BUFFER
08F2 EB                   XCHG
08F3 2A0810               LHLD    TBIP
                          INCTB   1
08F6+3E01                 MVI     A,1
08F8+D7                   RST     2
08F9 C8                   RZ      ; RETURN WITH ZERO STATUS IF OVERFLOW
08FA 220810               SHLD    TBIP    ; UPDATE POINTER IF OK
08FD 71                   MOV     M,C     ; MOVE DATA INTO BUFFER
08FE 211C10               LX1     H,BUFCOUNT      ; INCREMENT COUNT OF DATA
0901  34                  INR     M
0902  7E                  MOV     A,M
0903  FEFB                CPI     251      ; HAVE WE GOT 251 BYTES NOW?
0905  C9                  RET              ; ZERO STATUS IF TOO MANY BYTES

0906 3E00       RESTART:MVI       A,0      ; SET COUNT IN BUFFER TO ZERO
0908  321C10              STA     BUFCOUNT
090B 2A0410               LHLD    CTBIE    ; SET INPUT POINTER JUST BEFORE
                          INCTB   1        ; DATA AREA
090E+3E01                 MVI     A,1
0910+D7                   RST     2
0911  220810              SHLD    TBIP
0914  219008              LX1     H,RSTART         ; AND RESTART RECEIVER
0917  224A10              SHLD    RNEXT
091A C34309               JMP     EXIT

091D 324710     R1CRC:    STA     RCRC1    ; SAVE FIRST CRC CHARACTER
0920  212909              LX1     H,R2CRC ; NOW GET SECOND CRC CHARACTER
0923 224A10               SHLD    RNEXT
0926 C34309               JMP     EXIT

0929  324610    R2CRC:    STA     RCRC2    ; SAVE SECOND CRC CHARACTER
092C DB04                 IN      MCR      ; RESET REQUEST TO SEND
092E E6FD                 AN1     0FFH-RTS
0930  D304                OUT     MCR
0932  3A4310              LDA     INTFLAG        ; INDICATE RECEIVE ROUTINE
0935  E6FE                AN1     0FFH-RXBUSY    ; IS NOT ACTIVE
0937  324310              STA     INTFLAG
093A  214309              LX1     H,EXIT   ; IGNORE ALL RECEIVE INTERRUPTS
093D 224A10               SHLD    RNEXT
0940 C34309               JMP     EXIT
0943 Cl         EXIT:     POP     B
0944 D1                   POP     D
0945 El                   POP     H
0946 Fl                   POP     PSW
0947 FB                   EI
0948 C9                   RET
                ***********************************************************
                ; TRANSMIT INTERRUPT ROUTINES

0949 2A4C10     TXINT:    LHLD    TNEXT    ; DISPATCH ADDRESS IN TNEXT
094C E9                   PCHL

0940 3E16       TSTART: MVI       A,SYN    ; OUTPUT A SYN CHARACTER
094F D300                 OUT     THR
0951  215A09              LX1     H,TDLE1 ; NEXT SEND A DLE
0954   224C10             SHLD    TNEXT
0957  C34309              JMP     EXIT

095A 3E10       TDLE 1:   MVI     A,DLE    ; OUTPUT A DLE
095C D300                 OUT     THR
095E  216709              LXI     H,TSTX   ; NEXT OUTPUT START OF TEXT
0961  224C10              SHLD    TNEXT
0964  C34309              JMP     EXIT

0967 3E02       TSTX:     MVI     A,STX    ; OUTPUT START OF TEXT
```

```
0969 D300                 OUT     THR
096B 217409               LX1     H,TDATA ; NEXT FUNCTION HANDLES TEXT DATA
096E 224C10               SHLD    TNEXT
0971 C34309               JMP     EXIT

0974 211D10     TDATA:    LXI     H,OUTCOUNT      ; MORE DATA IN BUFFER?
0977  7E                  MOV     A,M
0978  B7                  ORA     A
0979 CA9709               JZ      TDATA1   ; NO, BUFFER EMPTY
097C 35                   DCR     M
097D 2A1A10               LHLD    LBOP
                          INCLB   1
0980+3E01                 MVI     A,1
0982+DF                   RST     3
0983  221A10              SHLD    LBOP     ; LBOP = LBOP+1
0986  7E                  MOV     A,M
0987  D300                OUT     THR      ; OUTPUT DATA AT LBOP
0989 FE10                 CPI     DLE      ; IS IT SAME AS DLE?
098B C24309               JNZ     EXIT     ; NO
098E 21A409               LXI     H,TDLE   ; TRANSMIT ANOTHER DLE
0991  224C10              SHLD    TNEXT    ; TO MAKE TRANSPARENT
0994  C34309              JMP     EXIT
0997  3E10      TDATA1: MVI       A,DLE    ; OUTPUT A DATA LINK ESCAPE
0999 D300                 OUT     THR
099B 21B109               LXI     H,TETX   ; NEXT SEND END OF TEXT
099E 224C10               SHLD    TNEXT
09A1 C34309               JMP     EXIT

09A4 3E10       TDLE:     MVI     A,DLE    ; SEND DATA LINK ESCAPE
09A6 D300                 OUT     THR
09A8  217409              LXI     H,TDATA ; AND GO BACK TO TRANSPARENT MODE
09AB 224C10               SHLD    TNEXT
09AE C34309               JMP     EXIT

09B1 3E03       TETX:     MVI     A,ETX    ; SEND END OF TEXT
09B3 D300                 OUT     THR
09B5 21BE09               LX1     H,T1CRC ; NEXT SEND FIRST CRC CHARACTER
09B8 224C10               SHLD    TNEXT
09BB C34309               JMP     EXIT

09BE 3A4910     T1CRC:    LDA     TCRC1    ; SEND FIRST CRC CHARACTER
09C1 D300                 OUT     THR
09C3 21CC09               LXI     H,T2CRC ; NEXT SEND SECOND CRC CHARACTER
09C6 224C10               SHLD    TNEXT
09C9 C34309               JMP     EXIT

09CC 3A4810     T2CRC:    LDA     TCRC2    ; SEND SECOND CRC CHARACTER
09CF D300                 OUT     THR
09D1 21DA09               LX1     H,TPAD   ; SEND TRAILING PAD CHARACTER
09D4  224C10              SHLD    TNEXT
09D7 C34309               JMP     EXIT

09DA 3EFF       TPAD:     MVI     A,PAD    ; SEND TRAILING PAD AFTER CRC
09DC D300                 OUT     THR
09DE 3A4310               LDA     INTFLAG ; MARK TRANSMIT NOT BUSY
09E1 E6FD                 AN1     0FFH-TXBUSY
09E3  324310              STA     INTFLAG
09E6 CD7608               CALL    DISABLETX
09E9 C34309               JMP     EXIT
                ***********************************************************
                ; CRC CALCULATION ROUTINE
                ; INCLUDES BYTE IN ACCUMULATOR IN CRC CALCULATION
09EC F5         CALCCRC:PUSH      PSW
09ED 0608                 MVI     B,8
09EF 4F                   MOV     C,A
09F0 2A4410               LHLD    CRC
09F3 =          CALCCRC1:EQU      $
09F3 79                   MOV     A,C
09F4 07                   RLC
09F5 4F                   MOV     C,A
09F6 7D                   MOV     A,L
09F7 17                   RAL
```

```
09F8 6F                 MOV     L,A
09F9 7C                 MOV     A,H
09FA 17                 RAL
09FB 67                 MOV     H,A
09FC D2070A             JNC     CALCCRC2
09FF 7C                 MOV     A,H
0A00 EE80               XRI     80H
0A02 67                 MOV     H,A
0A03 7D                 MOV     A,L
0A04 EE05               XRI     05H
0A06 6F                 MOV     L,A
0A07 =         CALCCRC2:EQU     $
0A07 05                 DCR     B
0A08 C2F309             JNZ     CALCCRC1
0A0B 224410             SHLD    CRC
0A0E F1                 POP     PSW
0A0F C9                 RET
               *****************************************************
               ; RECEIVE SIDE DISPATCH ROUTINES

0A10 2A4E10    DISPRX:  LHLD    RDISP    ; GO TO RECEIVE DISPATCH ROUTINE
0A13 E9                 PCHL

0A14 2A0610    WAITTB:  LHLD    OTBE     / TERMINAL BUFFER CUSHION FREE?
0A17 EB                 XCHG
0A18 2A0410             LHLD    CTBIE
                        COMPARE          ; COMPARE DE TO HL
0A1B+EF                 RST     5
0A1C CA280A             JZ      WAITTB1 ; SAME, BUFFER AVAILABLE
                        INCTB   CUSHION ; IS CUSHION FREE?
0A1F+3EFF               MVI     A,CUSHION
0A21+D7                 RST     2
0A22 DA120B             JC      DISPTX   ; TO TRANSMIT ROUTINE DISPATCHER
0A25 2A0410             LHLD    CTBIE    ; POINT TBIP JUST AHEAD OF DATA
               WAITTB1:INCTB    1
0A28+3E01               MVI     A,1
0A2A+D7                 RST     2
0A2B 220810             SHLD    TBIP
0A2E 3E00               MVI     A,0      / ZERO COUNT FOR RECEIVE ROUTINE
0A30 321210             STA     BUFCOUNT
0A33 F3                 DI
0A34 3A4310             LDA     INTFLAG ; RECEIVE ROUTINE IS ACTIVE
0A37 F601               OR1     RXBUSY
0A39 324310             STA     INTFLAG
0A3C FB                 EI
0A3D 219008             LX1     H,RSTART        ; START RECEIVING
0A40 224A10             SHLD    RNEXT
0A43 DB04               IN      MCR      / SET RTS SO OTHER END WILL SEND
0A45 F602               OR1     RTS
0A47 D304               OUT     MCR
0A49 21500A             LX1     H,WAITRX        ; WAIT FOR BLOCK
0A4C 224E10             SHLD    RDISP
0A4F C9                 RET

0A50 3A4310    WAITRX:  LDA     INTFLAG ; IS RECEIVER STILL BUSY?
0A53 E601               AN1     RXBUSY
0A55 C2120B             JNZ     DISPTX   ; YES, GO TO TRANSMIT DISPATCHER
0A58 3A5210             LDA     DFLAG    ; GET DISPATCHER FLAG
0A5B E601               AN1     CRCTX    ; IS CRC ROUTINE BUSY?
0A5D C2120B             JNZ     DISPTX   ; YES, GO TO TRANSMIT DISPATCHER
0A60 211C10             LX1     H,BUFCOUNT      ; COUNT OF BYTES RECEIVED
0A63 7E                 MOV     A,M
0A64 2A0410             LHLD    CTBIE    ; POINT TO CURRENT INPUT ENTRY
0A67 77                 MOV     M,A      ; PUT COUNT IN BUFFER HEADER
                        INCTB   1        ; POINT JUST BEFORE DATA AREA
0A68+3E01               MVI     A,1
0A6A+D7                 RST     2
0A6B 220810             SHLD    TBIP
0A6E 210000             LX1     H,0      ; INITIALIZE CRC VALUE
0A71 224410             SHLD    CRC
0A74 217B0A             LX1     H,RXCRC ; NEXT CALCULATE CRC
0A77 224310             SHLD    RDISP
```

```
0A7A C9                 RET

0A7B 211C10    RXCRC:   LXI     H,BUFCOUNT      ; MORE DATA TO INCLUDE?
0A7E 7E                 MOV     A,M
0A7F B7                 ORA     A
0A80 CA970A             JZ      RXCRC1   ; NO, GO TO INCLUDE CONTROL CHARS
0A83 35                 DCR     M        / DECREMENT COUNT
0A84 2A0810             LHLD    TBIP     / UPDATE POINTER TO NEXT POSITION
                        INCTB   1
0A87+3E01               MVI     A,1
0A89+D7                 RST     2
0A8A 220810             SHLD    TBIP
0A8D 7E                 MOV     A,M      ; GET DATA BYTE IN A
0A8E CDEC09             CALL    CALCCRC ; INCLUDE IT IN CRC CALCULATION
0A91 FE10               CPI     DLE      / WAS IT DATA LIKE A DLE?
0A93 CCEC09             CZ      CALCCRC ; DO ANOTHER FOR TRANSPARENCY
0A96 C9                 RET              / RETURN TO LIP
0A97 3E10      RXCRC1:  MVI     A,DLE    ; INCLUDE DLE AND ETX IN CRC
0A99 CDEC09             CALL    CALCCRC
0A9C 3E03               MVI     A,ETX
0A9E CDEC09             CALL    CALCCRC
0AA1 21A80A             LXI     H,CHECKCRc      , NEXT CHECK THE CRC
0AA4 224E10             SHLD    RDISP    ; GO INCLUDE CRC CHARS RECEIVED
oAA7 C9                 RET

0AA8 3A4710    CHECKCRC:LDA     RCRC1    ; INCLUDE RECEIVED CRC CHARACTERS
0AAB CDEC09             CALL    CALCCRC
0AAE 3A4610             LDA     RCRC2
0AF31 CDEC09            CALL    CALCCRC
0AB4 21BB0A             LXI     H,CHKFIN        ; NEXT CHECK IF CRC IS GOOD
0AB7 224310             SHLD    RDISP
0ABA C9                 RET

0ABB 2A4410    CHKFIN:  LHLD    CRC      ; GET CALCULATED CRC
0ABE 7D                 MOV     A,L      ; IS IT ZERO?
0ABF B4                 ORA     H
0AC0 C2D00A             JNZ     CHKFIN1 ; NO, GO RESTART RECEIVE OPERATION
0AC3 DB04               IN      MCR      ; YES, GOOD CRC, FLIP DTR
0AC5 EE01               XRI     DTR
0AC7 D304               OUT     MCR
0AC9 21D70A             LXI     H,RPROC ; PROCESS CHECKED BLOCK
0ACC 224310             SHLD    RDISP
0ACF C9                 RET
0AD0 21140A    CHKFIN1:LXI      H,WAITTB        ; BAD CRC, TRY AGAIN
0AD3 224E10             SHLD    RDISP
0AD6 C9                 RET

               ; THIS ROUTINE SHOULD PROCESS THE BUFFER PREFIX
               ; TEMPORARILY IT ONLY PASSES THE BUFFER TO THE LIP
               ; AND HANDLES CONNECT/DISCONNECT
0AD7 2A0410    RPROC:   LHLD    CTBIE    ; IS COUNT 7 OR MORE?
0ADA 7E                 MOV     A,M
0ADB FE07               CPI     7
0ADD DA020B             JC      RPROC2   ; NO, PASS TO LIP
                        INCTB   8        / POINT TO SEE IF CONNECT OR
0AE0+3E08               MVI     A,8
0AE2+D7                 RST     2
0AE3 7E                 MOV     A,M      / DISCONNECT
0AE4 FE01               CPI     'A'-40H ; IS IT CONNECT?
0AE6 C2F30A             JNZ     RPROC1   ; NO, GO TO TEST FOR DISCONNECT
0AE9 3E00               MVI     A,0      i 0 FOR CONNECT
0AEB F7                 RST     6        ; COMMUNICATE REQUEST TO LIP
0AEC 21140A             LXI     H,WAITTB        ; DON'T PASS THIS ENTRY
0AEF 224E10             SHLD    RDISP
0AF2 C9                 RET
0AF3 FE02      RPROC1:  CPI     'B'-40H ; IS IT DISCONNECT?
0AF5 C2020B             JNZ     RPROC2   ; NO, PASS TO LIP
0AF8 3E01               MVI     A,1      ; YES, 1 FOR DISCONNECT
0AFA F7                 RST     6        ; COMMUNICATE REQUEST TO LIP
0AHH 21140A             LXI     H,WAITTB        ; DON'T PASS THIS ENTRY
0AFE 224310             SHLD    RDISP
0B01 C9                 RET
```

```
0B02 2A0810      RPROC2: LHLD    TBIP     ; UPDATE CURRENT INPUT ENTRY
                         INCTB   1
0B05+3E01                MVI     A,1
0B07+D7                  RST     2
0B08  220410             SHLD    CTBIE
0B0B 21140A              LX1     H,WAITTB         ; NOW GO GET ANOTHER ONE
OBOE 224E10              SHLD    RDISP
0B11 C9                  RET
                 ***********************************************************
                 ; TRANSMIT SIDE DISPATCH ROUTINES

0B12 2A5010      DISPTX: LHLD    TDISP    ; GO TO TRANSMIT DISPATCH ROUTINE
0B15 E9                  PCHL

                 ; THIS ROUTINE SHOULD PROCESS THE BUFFER PREFIX BUT
                 ; TEMPORARILY IT ONLY PASSES THE BUFFER TO THE HOST
                 WAITLIP:NEXTIN  ; IS THERE A BUFFER ENTRY FROM THE LIP?
0B16+E7                  RST     4
0B17 C8                  RZ               ; NO, RETURN
0B18 3A5210              LDA     DFLAG    ; INDICATE TX SIDE USING CRC
0B1B F601                OR1     CRCTX    ; ROUTINES
0B1D  325210             STA     DFLAG
0B20 7E                  MOV     A,M      ; GET DATA LENGTH FROM HEADER
0B21 321D10              STA     OUTCOUNT ; FOR CRC CALCULATION ROUTINE
                         INCLB   3        ; POINT JUST BEFORE DATA AREA
0B24+3E03                MVI     A,3
0B26+DF                  RST     3
0B27 221A10              SHLD    LBOP     ; FOR CRC CALCULATION ROUTINE
0B2A 210000              LX1     H,0      ; INITIALIZE CRC VALUE
0B2D  224410             SHLD    CRC
0B30 21370B              LX1     H,TXCRC  ; NEXT START CRC CALCULATION
0B33  225010             SHLD    TDISP
OB36  C9                 RET

0B37 211D10      TXCRC:  LXI     H,OUTCOUNT     ; ANY MORE DATA TO INCLUDE?
0B3A 7E                  MOV     A,M
0B3B B7                  ORA     A
0B3C CA530B              JZ      TXCRC1   ; NO, GO TO INCLUDE CONTROL CHARS
0B3F 35                  DCR     M        ; DECREMENT COUNT
OB40 2A1A10              LHLD    LBOP     ; UPDATE POINTER TO NEXT POSITION
                         INCLB   1
0B43+3E01                MVI     A,1
0B45+DF                  RST     3
0B46 221A10              SHLD    LBOP
0B49 7E                  MOV     A,M      ; GET DATA BYTE IN A
0B4A CDEC09              CALL    CALCCRC ; INCLUDE IT IN CRC CALCULATION
0B4D FE10                CPI     DLE      ; WAS IT DATA LIKE A DLE?
0B4F CCEC09              CZ      CALCCRC ; DO ANOTHER FOR TRANSPARENCY
0B52 C9                  RET              ; RETURN TO LIP
0B53 3E10        TXCRC1: MVI     A,DLE    ; INCLUDE DLE AND ETX IN CRC
0B55 CDEC09              CALL    CALCCRC
0B58 3E03                MVI     A,ETX
0B5A CDEC09              CALL    CALCCRC
0B5D 21640B              LXI     H,CRCFIN ; NEXT TO FINISH CRC FOR SENDING
0B60 225010              SHLD    TDISP
0B63 C9                  RET

0B64 3E00        CRCFIN: MVI     A,0       ; FINISH OFF CRC CALCULATION FOR
OB66 CDEC09              CALL    CALCCRC   ; TRANSMISSION
0B69 CDEC09              CALL    CALCCRC
0B6C 2A4410              LHLD    CRC       ; SAVE CALCULATION FOR TRANSMIT
0B6F  224810             SHLD    TCRC2
0B72 21790B              LX1     H,STARTTX        ; NEXT, START TRANSMITTING
0B75 225010              SHLD    TDISP            ; THE BLOCK
0B78 C9                  RET

0B79 3A4210      STARTTX:LDA     MSRSAVE  ; SAVE CURRENT DSR LEVEL
0B7C E620                AN1     DSR
0B7E 67                  MOV     H,A
0B7F 3A5210              LDA     DFLAG
0B82 E6DE                AN1     0FFH-CRCTX-DSR   ; INDICATE CRC ROUTINE
OB84 B4                  ORA     H                ; NOT IN USE AND SAVE
```

```
0B85 325210              STA     DFLAG            ; DSR IN DFLAG
OB88 214D09              LX1     H,TSTART         ; SET UP INITIAL XMIT
0B8B 224C10              SHLD    TNEXT            , INTERRUPT ROUTINE
0B8E 2A1610              LHLD    OLBE     ; POINT TO DATA TO TRANSMIT
0B91 7E                  MOV     A,M      ; GET COUNT
0B92 321D10              STA     OUTCOUNT
                         INCLB   3
0B95+3E03                MVI     A,3
0B97+DF                  RST     3
OB98 221A10              SHLD    LBOP
0B9B F3                  DI               ; DISABLE INTERRUPTS
0B9C 3A4310              LDA     INTFLAG ; INDICATE TRANSMIT BUSY
0B9F F602                OR1     TXBUSY
0BA1 324310              STA     INTFLAG
0BA4 3A4210              LDA     MSRSAVE ; IS CTS UP?
0BA7 E610                AN1     CTS
0BA9 CAB20B              JZ      STARTTX1 ; DON'T ENABLE TRANSMIT INTRPT
OBAC DB01                IN      IER      ; YES, ENABLE TRANSMIT INTERRUPTS
OBAE  F602               OR1     ETBEI
0BB0 D301                OUT     IER
0BB2 FB         STARTTX1:EI               ; ENABLE INTERRUPTS
0BB3 21BA0B              LX1     H,WAITTX ; WAIT FOR TRANSMIT TO FINISH
OBB6 225010              SHLD    TDISP
0BB9 C9                  RET

OBBA 3A4310     WAITTX:  LDA     INTFLAG ; TRANSMITTER INTERRUPTS ENABLED?
OBBD E602                ANI     TXBUSY
OBBF C0                  RNZ              ; YES, RETURN
0BC0 211027              LX1     H,ACKTO ; NO, SET UP FOR TIMEOUT
0BC3 224010              SHLD    WAIT     ; INITIALIZE ACKNOWLEDGE TIMEOUT
OBC6 21CDOB              LX1     H,WAITACK      ; NEXT WAIT FOR ACKNOWLEDGE
0BC9 225010              SHLD    TDISP
OBCC C9                  RET

                WAITACK:
OBCD 215210              LX1     H,DFLAG ; IS DSR SAME AS BEFORE?
OBDO 3A4210              LDA     MSRSAVE
0BD3 AE                  XRA     M
0BD4 E620                ANI     DSR
0BD6 C2EA0B              JNZ     WAITACK1          ; NO, BLOCK ACKNOWLEDGED
0BD9 2A4010              LHLD    WAIT     ; YES, DECREMENT TIMEOUT COUNT
OBDC 2B                  DCX     H
0BDD 224010              SHLD    WAIT
0BE0 7C                  MOV     A,H      ; IS TIME OVER?
0BE1 B5                  ORA     L
0BE2 C0                  RNZ              ; NO, RETURN
0BE3 217908              LX1     H,STARTTX ; YES, TIMED OUT, SO SEND AGAIN
0BE6 225010              SHLD    TDISP
0BE9 C9                  RET
                WAITACK1:
0BEA 21160B              LXI     H,WAITLIP  ; GOOD ACK, GET ANOTHER BUFFER
0BED 225010              SHLD    TDISP      ; FROM LIP
0BF0 C9                  RET

0BF1                     END
```