MITRE-Bedford

MTR-4729

JSC #16054

The NASA Bus Communications Listening Device Software

M. A. Allen

JULY 1979

MITRE

MITRE Technical Report
MTR-4729

The NASA Bus Communications Listening Device Software

M. A. Allen

JULY 1979

CONTRACT SPONSOR	NASA/JSC
CONTRACT NO.	F19628-79-C-0001; T5295F
PROJECT NO.	8470
DEPT.	D-74

THE MITRE CORPORATION
HOUSTON, TEXAS

ABSTRACT

The need for an inexpensive way to monitor the two-way traffic on the prototype MITRE bus communications cable system at the National Aeronautics and Space Administration Johnson Space Center prompted the development of the bus listener described in this document. This report is intended to serve as a user's guide for the bus listener as well as document the code used in the Bus Interface Unit (BIU). For Bedford users, the source code resides in TSO account 770 under the name LISTEN.ASM.

TABLE OF CONTENTS

Table of Contents
(Continued)

Table of Contents
(Concluded)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I
INTRODUCTION

1.0 BACKGROUND

MITRE was contracted by the National Aeronautics and Space Administration to provide a prototype Bus Communications System to connect the host Modular Computer Systems MODCOMP IV of the Trend Monitoring System (TMS) to several MEGATEK graphics display terminals. Special software was developed to control the Bus Interface Units (BIU's) connecting each of these devices to the MITRE communications cable to form the bus communications network.

The software development involved extensive modifications to existing software which was operational at MITRE's home office in Bedford, Massachusetts. The primary modifications involved the development of a Direct Memory Access (DMA), parallel interface to the MODCOMP and MEGATEK ports. Modification of the original code and addition of new code led to a long debugging process. During this test and validation process, it was determined that the ability to examine data packets as they were transmitted would be very helpful in establishing which element of the network was the cause of any particular difficulty. It was determined that the most economical approach to provide this capability would be to use an existing serial terminal device, such as a teletype or CRT, attached to the cable through a standard serial BIU

with special software operating in the BIU. This code would effectively listen to the bus to monitor any or all addresses on the network and print the data addressed to the selected devices on the display.

1.1 Scope

This report is intended to serve as a user's guide for anyone wishing to employ the bus listener BIU to monitor network traffic using a serial terminal device. This monitoring may be required to troubleshoot new or existing network applications as described above. Any terminal device, capable of interfacing with a serial BIU via an RS-232C cable connector, can serve as an output device.

Section II is a basic discussion of the architecture employed by the MITRE Bus Communications System and how the listener will interact with it. Section III is intended as a user's guide to operation of the bus listener. Section IV is a detailed discussion of the software modifications made to the standard NASA serial terminal BIU code. Appendix I and II contain the software listings and flow charts respectively.

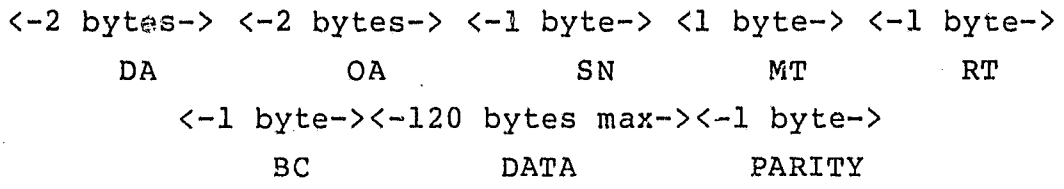
SECTION II
THE BUS COMMUNICATIONS SYSTEM

2.0 INTRODUCTION

This section is intended to provide the reader with a basic understanding of the functioning of a BIU and its relation with the communications network. It is not intended to provide the reader with a detailed understanding of the system. A basic understanding of the network topology is assumed. For furher information on the communications system architecture and the system software, the reader is directed to MTR-4721 (JSC #14723), "TMS Communications Hardware - Volume II - Bus Interface Unit" and MTR-4723 (JSC #14793), "Trend Monitoring Systm (TMS) Communications Software - Volume II - Bus Interface Unit (BIU) Software."

2.1 Protocol

The MITRE bus communications system uses an unslotted, carrier-sense multiple access discipline which employs a contention "listen-while-talk" (LWT) protocol for network control. Data is packetized for transmission with header information attached in accordance with the diagram in Figure 2.1 below:



DA = Destination Address
OA = Originator's Address
SN = Sequence Number
MT = Message Type
RT = Retry Count
BC = Byte Count (8 bits per byte)
PARITY = Longitudinal Parity Byte
(Not included in byte count)

Figure 2.1: LWT Bus Packet Format

2.1.1 Listen-While-Talk

When a unit has data to be transmitted on the cable, it monitors the bus for the presence of the communications carrier. If the carrier is present, the BIU is inhibited from entering the transmit mode. In addition, when a particular unit enters the transmit mode, it monitors its own transmission to determine if a data byte was garbled by noise or another BIU transmitting on the network. These collisions can result from the finite time delay necessary for the data to circumnavigate the cable. Both BIU's involved in the collision will detect the error and stop transmitting for a random amount of time whereupon a retransmission of the last data packet

would be attempted. Since a random backoff based on a device's home address is used, the possibility of mutual deadlock is avoided. This protocol allows the theoretical system throughput to approach 99% of available bandwidth when the data requirements of the user are "bursty" in nature, that is, the data entries are relatively short in length and randomly distributed in time.

2.1.2 Bus Addressing

The protocol calls for the first two bytes of each network packet to be the address of the device which should receive and process the packet. These first bytes will cause each BIU on the network to register an internal interrupt when its receive data register becomes full. As a result of this interrupt, a comparison will be made to a home address maintained in each BIU (Note: All current applications use only the first byte of the address). If the data packet is not addressed to the BIU, the BIU's receiver will be disabled until a Non-Maskable Interrupt (NMI) occurs as a result of the communications carrier being dropped by the transmitting BIU. In this manner, all BIU's which are not interested in the data packet (i.e. not addressed) will continue with other processing and avoid being continually interrupted by unwanted data.

2.1.3 Message Continuity

The message header uses three values to enable the continuity of data packets to be maintained. Each received packet is checked to determine who originated the packet, whether the packet was received in sequence, and

whether the packet was transmitted more than once. These values of originator's address, sequence number and retry count can be stored and compared to previous values to determine if message continuity has been preserved. Should an unexpected value be detected, the packet can be ignored or handled in a special manner depending on the application.

2.1.4 Message Types

Provision has been made to have the BIU recognize special message types. These special messages include sign-on, sign-off, and status messages. Other types of messages can be used depending on the application. A special code, defined by the application developer to indicate message types, is inserted into each packet.

2.2 Specifications

The efficiency of the protocol is based on the ratio of the packet length to the total propagation delay. In the TMS system a data transmission rate of 307.2 kilobits/second is used with a maximum packet length of 128 bytes and a minimum packet length of 8 bytes. The maximum length of the communications cable is determined by the time necessary for the first two address bytes to be transmitted on the newtwork. This creates a time window within which a collision, caused by another BIU starting transmission, will be detected. The twenty bits of the address bytes (including start and stop bits) limit the maximum cable length to approximately 10 miles.

2.3 Error Detection and Correction

The addition of checksums and packet acknowledgments provides the capability of error detection and correction. Each packet contains one byte of longitudinal parity formed by the exclusive ORing of all other data bytes in the packet. If this checksum is received in error, the addressee will ignore the packet and thus cause the sender to retransmit the packet. This retransmission occurs automatically when the sender does not receive an acknowledgment of the last transmitted packet within the acknowledgment time window of 100 microseconds.

2.4 BIU Hardware

The basic digital logic used in the NASA BIU is designed around a MOS Techology 6502A microprocessor. It uses 3072 bytes of random access memory (RAM) with 2048 bytes of programmable read only memory (PROM). The RAM is used for the storage of several variable flags and pointers and the storage of the packetized data buffers being queued for transmission on the network or received from the network. The PROM contains the main BIU operational software. Communication to and from the BIU is accomplished by means of two Motorola 6850 Asynchronous Communications Interface Adapters and two MOS Technology 6522 Versatile Interface Adapaters (VIA's). In addition to the parallel interface capability it provides, each VIA also contains two timers which can be used as system clocks. Some variations of the BIU hardware use differing numbers of these basic chips but the general architecture remains the same.

2.5 BIU Listener

A basic serial BIU is used to implement the bus listener described below. It takes advantage of the normal addressing scheme of the BIU's to eavesdrop on their communications. Depending on the parameters supplied by the user, the listener will monitor one, two or all addresses on the network. When it detects a valid address, it stores the following data packet in its buffer and decrements the packet count. Acknowledgements are treated as packets and will likewise decrement the packet count. When the packet count goes to zero, the listener dumps its memory to the terminal device. The user interface is described in the following section.

SECTION III

BUS LISTENER OPERATING PROCEDURES

3.0 INTRODUCTION

The following paragraphs may be used as an operator's guide to the use of the bus listener described in Section IV. The description of the dialog between the user and the device will be of use in understanding the software and the terms used in the software description.

3.1 Hardware Set Up

To operate the BIU Listener it is necessary to use a standard serial terminal BIU with the special bus listener PROM's installed. This BIU may then be connected to any serial terminal device through the RS232C connector at the rear of the BIU. It is then necessary to connect the receive MODEM of the BIU to the communications cable. Since the bus listener never originates any transmissions on the network, it is not necessary to connect the transmit MODEM. Messages received by the listener are never acknowledged.

The data rate for the serial terminal should be set in the normal manner using the Dual Inline Package (DIP) switches on the BIU's digital board. For further information on this function, the reader is directed to MTR-4724 (JSC #14794), "Diagnostic Procedures for Trend Monitoring System (TMS) Communications." For terminals with mechanical carriage control (e.g. Texas Instruments

Silent 700®), a special null routine has been added to allow time for the carriage to return to the start of each line.

3.2 Operator Procedures

With the bus listener connected to the cable as described above the power should be turned on. As a result of the power up sequence the terminal device will display the initial message for operation of the listener. The BIU is now in a tight loop waiting for the operator's response to a series of these initialization messages. The series of messages and some valid responses are shown in the example below:

PACKET COUNT? (00 - FF, HEX)
10 (requesting 16 packets)

PACKETS SKIPPED? (00 - FF, HEX)
0 (requesting that no packets be skipped)

MONITOR ADDRESS? (HEX)
BB (requesting that messages to the backboard be monitored)

OPTIONAL SECOND ADDRESS? (HEX)
41 (Requesting that messages to TERMA be monitored)

It can be seen from the example above that the operator can monitor one or two specific addresses on the network. It is assumed that the operator has prior information concerning valid addresses. This information can come from an intimate knowledge of the network architecture or may be obtained from a listing of the data monitored by a network technical controller or status recorder. If this information is not available, it is possible to enter the command 'ALL' in response to the MONITOR ADDRESS or OPTIONAL SECOND ADDRESS requests. This will result in the printing of the specified number of packets transmitted on the network regardless of the addresses. By examining the addresses of these data packets (see paragraphs 2.1 and 4.4.2.1), it is possible to obtain valid device codes for future listening operations.

3.3 Multiple Address Monitoring

The two-address option is very useful in observing two-way communications between nodes of the network. Since the listener keys on the first packet of each transmission, the listener records data transmitted and then the acknowledgment by the addressee. Each acknowledgment counts as a data packet for packet counting purposes. In this manner it is possible to determine if a BIU has received the data and is acknowledging properly.

3.4 Skipping Packets

The ability to skip a given number of packets before initiating the recording of data allows the

capturing of packets at the end or in the middle of very long packet streams. By entering something other than zero for this value, that number of packets will be skipped before the packet count starts to decrement. This value of skipped packets is decremented based on the addressing information given in the monitor address commands to insure only packets with valid addresses affect the skip count.

3.5 Maximum Packet Count

The maximum number of packets that can be presented on any one operation is 22 for BIU's with 3K of RAM available. Since the bus listener uses a circular buffer operation, the last 22 packets received will be displayed. This means that packet counts larger than 16 hex will result in the loss of the first N-22 data packets.

3.6 Interrupting Processing

At any time during the process of recording packets, the procedure may be interrupted and the packets recorded to that point displayed at the terminal. This is accomplished by pressing the ESC key (hex code 1B) on the terminal device. This function is useful if the operator would like to look at some intermediate results or if an error was made in entering the monitor address and no apparent recording is taking place. It is also useful to terminate an infinite recording operation.

3.7 Infinite Processing

By entering zero for the packet count, the recording of data packets on the network will continue until halted by the operator. The listener will not count packets and will continue to record all transmissions with the proper address until the ESC key is entered. The pressing of the ESC key will result in the printing of the PROCESSING INTERRUPTED message on the terminal followed by the captured data and a new initialization message.

3.8 Normal Termination

When the packet count requested is reached, the terminal will print the PROCESSING FINISHED message followed by the data packets and then the initialization message. This indicates that all processing followed the normal procedure.

3.9 Error Messages

If, in the course of entering commands to the listener or during the printing of data packets, an error is detected, an appropriate error message will be printed and the initialization message presented again. The error message for key errors is INPUT ERROR, TRY AGAIN. If an error is detected during the processing of data packets, the message ABNORMAL ENDING ERROR will be displayed (see Error Recovery below).

3.10 Error Recovery

For most errors, the system will return to the reset procedure and print out the first initialization message. Should this fail to occur or should the system become inexplicably hung, the user can usually regain control of the listener by pressing the RESET button on the front panel of the BIU; however, this action will cause the loss of any data recorded to this point. If it is desirable to obtain this data, it would be better to try entering an ESC character before pressing RESET.

SECTION IV

THE BUS LISTENER SOFTWARE

4.0 INTRODUCTION

This section describes the software used in the BIU's PROM to implement the bus listener. The basic format of the code is derived from the software which was designed for the NASA serial terminal BIU. Several modifications are made to this basic software and are detailed below. The modified routines include INDEV, OUTDEV, ENQ, and IRQ, in addition, several routines are completely eliminated. The deleted routines include NET, CHKOUT, STIMER, PCONST, and SFINC. The primary reason for the elimination of these routines is the fact that the bus listener is not required to transmit any data on the network.

4.1 Approach to Development

It was determined that the best approach to programming the bus listener BIU would be to use existing code for the NASA terminal BIU and adapt it for the special purpose described above. As cited above, several routines are eliminated and several others modified. In addition, other routines are expanded to allow for special processing.

4.1.1 Circular Buffering

It was determined that the best approach to the storing of data packets is to use a circular buffer instead of the first-in-first-out (FIFO) buffer normally programmed for use in the BIU. The driving factor for this change is the need to save an unknown number of packets with a limited buffer space (see Infinite Count below) and thus the need to overwrite buffers continually. A side effect of the FIFO buffer scheme which was never encountered in normal processing is the long time delay induced by the search through the queue for the next available buffer (see Time Criticality below). This delay results from having only the starting address of a queue and having to go from buffer pointer to buffer pointer looking for the last buffer indicator. This delay, which comes into play during the recording of acknowledgments, is virtually eliminated by the circular scheme.

4.1.2 Listen Operation

To accomplish the monitoring operation, the IRQ routine was modified in the following manner. With the variable ALFLG set by the "ALL" response during the initialization dialog, address checking is suspended and each packet received from the network is stored in a buffer ready to be queued. If ALFLG is not set, the address of each received packet is compared to the two addresses stored in HOME1 and HOME2 during the initialization sequence. If either address is found, the data packet is stored ready to be queued pending the outcome of the packet skip and packet count checks.

Regardless of the condition of ALFLG, the data will not be queued if the packet skip count is still greater than zero; however, the count will be decremented if the address is valid. If the packet count is greater than zero, the packet will be queued and the count decremented if the address is valid. Once the packet count reaches zero, the interrupt mask should be set to prevent any more packets from halting output processing, however, one or more packets may be detected before this mask can be set. In this event, the packets will be ignored (not queued) and control will be returned to the main program to allow output processing. No acknowledgments are ever sent since the listener is not the packet addressee.

4.2 Reset Operation

The primary function of the RESET procedure is maintained in the new software. This routine still initializes all page zero variables, sets all buffer pointers, and initializes the communications interface chips.

4.2.1 Stack Pointer

Stack pointer initialization is modified to take advantage of unused space in the page one area of memory to enable the creation of an additional buffer. It was determined that to restrict the processor stack to operation between memory locations 017F and 0100 would have no impact on system operation. For this reason the stack pointer is initialized to 7F. (Note: The value 01

is assumed on all stack operations for the high order byte of the two-byte address.) The area of page one from 01FF through 0180 is then used for the second buffer (see paragraph 4.2.3 below).

4.2.2 Page Zero Variables

The page zero variables start at memory loation 0000 and go to the end of the buffer pointers in BUFSTK. The variables that need to be initialized to zero are located below CONECT and reset by a tight loop which uses the address of CONECT as the starting index. The reset procedure reflects this plan and allows for future expansion if necessary. It should be pointed out that the final page zero variable currently occurs at memory loaction 0076.

4.2.3 Buffer Pointers

It was determined that it is useful to squeeze all the buffer space available out of the RAM. For this reason the gap between the end of the page zero variables and the bottom of the stack is used as the first buffer. During the RESET procedure the first buffer address is set to 0080. The end of this first buffer is therefore at address 00FF. The remaining 21 buffer addresses are set in the buffer initialization loop starting with buffer 2 at 0180 and ending with buffer 22 at loaction 0B80. This buffer assignment leads to the map of memory shown in Table I below.

Table I
Memory Map of Bus Listener RAM

0000	-	007F	Zero Page Variables & Pointers
0080	-	00FF	Buffer #1
0100	-	017F	Processor Stack
0180	-	0BFF	Buffers #2 - #22

4.2.4 Initialization Messages

The final modification made to the RESET procedure involves the addition of code to carry on a dialog with the user. This process results in the setting of several flags and counters to enable the recording of data packets to proceed automatically. After each call to the PTSTR routine (see paragraph 4.8), the message indexed by the Y register will be displayed at the terminal and a jump to the INDEV subroutine executed. The INDEV routine is used to process user input from the terminal.

4.3 INDEV Subroutine

Several changes to the INDEV routine are needed to allow the bus listener dialog to proceed. No messages are sent on the bus so the portion of the routine dedicated to packetizing the terminal input is removed. In a like manner, no other BIU will attempt to sign-on to the listener so the code involved with a sign-on response is deleted. The remaining code concerned with a reply to the WHICH SYSTEM? request is modified to accept proper input responses to the initializing messages.

4.3.1 Valid Inputs

Three types of input from the terminal are allowed by the INDEV routine: hexadecimal inputs in response to initialization messages, the phrase "ALL" in response to the address initialization messages, and the ESC character during normal processing. The CONECT flag is used to distinguish between two of the states. With the CONECT flag set to a negative value, the code assumes inputs are in response to initialization messages; if the flag is positive (zero), the code assumes there is an input during the normal recording process. In the latter case the input is examined for an ESC character to indicate the desire to terminate the recording process and print the buffered data. Any other entry by the operator is ignored.

With the CONECT flag set to a negative value, inputs are stored until a carriage return is detected or three characters are entered. Once either of these two cues is detected, the stored input is compared against a table of valid hexadecimal characters. If the entries are not valid hex characters, a comparison is made against the alphabetic string "ALL" but only if this is the third or fourth pass through the INDEV routine and an address to be monitored is expected. If there is still an error, then the input error message is displayed at the terminal.

4.3.2 Matching String

If all the characters are hexadecimal or equate to the string "ALL," then the code branches to MATCH for hex characters or MATCHA for "ALL." In MATCH a counter records the number of valid responses to control the channelling of the data to the proper flag or counter. After the fourth pass through MATCH or a pass through MATCHA has occurred, the program is ready to begin listening and control is passed to the main loop.

4.3.3 Setting Counters

Since the input string is in ASCII format, there must be a conversion made to allow the setting of a binary counter. This occurs in two subroutines called ALPHA and CTON. ALPHA checks to see if the hex digit entered is one of the six possible alphabetic characters. If it is, a weighting factor is used during the character-to-numeral conversion in CTON. CTON also takes into consideration the relative positions of the characters in the input string to allow proper scaling of the counter. The maximum value allowed in any counter is 256 (hex FF). CTON also insures that this limit is not violated. If this value is exceeded, the code will print the input error message on the terminal and return to RESET and the initialization routine.

4.3.4 Infinite Count

Provision is made in INDEV to allow the operator to indicate that he wishes the recording of data packets to continue indefinitely. On the first pass through INDEV, if the value zero is entered for the PACKET COUNT, it is interpreted to indicate the request for indefinite data recording and the FLEET flag is set. It should be pointed out that a carriage return without any input at this point will result in the same interpretation. To stop the recording of data packets, the operator enters ESC during the normal operating cycle to output any packets that were recorded (last 22 if more than this maximum were received) or push the BIU RESET button to start over again.

4.3.5 Monitor All Addresses

If the input during the third or fourth pass through INDEV is decoded to indicate a request for all addresses (ALL), the code branches to the MATCHA routine. This routine sets a flag (ALFLG) to indicate that the address comparison portion of the IRQ routine should be bypassed. This routine terminates by passing control to the READY routine, thus eliminating any unnecessary dialog.

4.3.6 Default Second Address

When a response other than "ALL" is made to the first request for a monitor address, a second address

request message is printed at the terminal. If the user ignores this request by entering only a carriage return, the program stores the value of the first address in the second address check variable. This will insure that only the one address is monitored. (Note: Entering only a carriage return in response to the first address request is an illegal response and will result in an error message and return to the RESET routine).

4.4 OUTDEV Subroutine

Changes to OUTDEV routine are required to insure that an easily understood data format is presented to the user. In addition, sections of the old OUTDEV routine used in interpreting non-data message types are eliminated, and special code is added to handle the unqueuing of the circular buffer.

4.4.1 Queue Processing

Since it is not advisable for the bus listener to miss any data packets of interest, no attempt is made to output data while the BIU is actively listening to the bus. Once the message count is fulfilled, the BIU's internal interrupt mask is set to prevent data packets on the bus from interrupting the continuous output of the stored packets. The queuing scheme used is a circular buffer. The method used to queue these buffers in the ENQ subroutine will lead to the first buffer being out of place if more than 21 packets are stored. For this reason the variable WRPFLG is set in the ALLOC subroutine to indicate this condition. If WRPFLG is non-zero, a

shifting of the buffer pointers takes place using QSTART and QEND to insure the data packets will be in proper order.

4.4.2 Data Output

Outputting characters to the terminal device is accomplished in the same manner as the old OUTDEV routine; however, two additional features are added. It is necessary to convert the binary data received from the bus into a presentable format for the display. Second, code is added to limit the number of characters presented on a line for ease of interpretation.

4.4.2.1 Data Conversion. Data characters, as received from the network, are in binary format. To enable the display of this information on a terminal device sensitive to control characters, the data are converted into ASCII format. This conversion results in the transformation of each 8 bit data byte into two hexadecimal characters. For clarity, a space is inserted between each byte of the packet. A typical BIU status message is presented below as an example.

```
00 00 41 00 01 DB 00 29 00 03 00 00 00 01 00 00
00 03 00 00 00 00 00 01 01 02 83 0C 24 02 54 45
52 40 49 4E 41 4C 20 42 49 55
```

As can be seen from the example this is terminal "A" reporting its status to address "00" which would normally be the status recorder. Interpretation of the last 12 bytes shows that this BIU is identifying itself as a "TERMINAL BIU."

4.4.2.2 Line Control. As mentioned above, it was determined to limit the number of bytes displayed per line. A variable LINCNT is incremented after each byte is output. After 16 bytes are presented the counter is reset to zero and a carriage return and line feed are output. An extra carriage return and line feed are output after each packet to improve packet definition.

4.4.2.3 Null String. A problem is encountered when a device with a mechanical carriage control is used as the output terminal. A finite amount of time is required for this type of device to position the carriage at the lefthand side of the page. This usually results in the loss of several data characters after a carriage return. To overcome this difficulty, a special PTSTR routine is used to precede each line of output with six "null" characters. The shift out and shift in ASCII characters (SO & SI, or Hex OE & OF) are selected for this purpose to insure minimum impact on user programs. Hex 00 or NULL could not be used since the standard BIU code uses this character to signal the end of a terminal message.

4.5 IRQ Subroutine

Only a few minor modifications are made to the IRQ routine. Since the listener never transmits data on the bus, it is not necessary to retain the code associated with the detecting of collisions. In a like manner, code associated with the transmission of acknowledgments is eliminated. A section of code is added to allow the monitoring of all addresses or any two addresses on the network.

4.5.1 Address Comparison

IRQ is entered as a result of the receive data register of the network UART becoming full and registering an interrupt. The first bytes of a packet are always the address of the BIU which should receive the packet (To Address). For this reason, each BIU on the network examines these first bytes to determine if the packet is meant for it. The listener will also examine this packet if ALFLG is not set to one. Unlike other BIU's, however, the listener will compare the address to two home addresses. If neither address matches, the receiver is disabled until a Non-Maskable Interrupt is registered.

4.5.2 Acknowledgments

Special code is added to allow the recording of acknowledgments. Normally, ACK's are ignored by all BIU's except for the one expecting it. To enable the listener to recognize these one-byte packets, however, it is necessary to keep normal processing path active when the transmit key falls. When an ACK is processed, the NMI resulting from the ACK occurs before the packet can be processed. Normally the data buffer being full but the transmission key being off would signal the end of processing and the code for turning off the receiver would be executed. Since the NMI (note: the NMI reactivates the receiver) has already occurred, this chain of events will leave the receiver turned off and cause the missing of the next packet. To avoid this problem, a check is made to see how many bytes were processed. If only one byte was received, the assumption is that it was an

acknowledgment and should be processed accordingly. The receiver is not disabled in this case and processing continues normally.

4.5.3 Time Criticality of Acknowledgment

The standard buffer mechanism of the serial BIU software performs a search for the end of the queue by starting at the beginning each time a buffer packet is enqueued. During this search interrupts are disabled to insure the packet being queued will not be lost. The time required for this search depends on the length of the queue. When approximately ten packets are queued, the time required exceeds the window allowed for the transmission of an acknowledgment. Since it is required that the acknowledgments be treated like any standard packet, the interrupt from the ACK will not be registered after the tenth packet is queued. This difficulty is overcome by the use of the special circular queue process. The circular queue is implemented with the NEXT pointers initialized to reflect which buffer follows the one being queued. The QSTART pointer is initialized to point to the first buffer and the QEND pointer is initialized to point to the last buffer. As buffers are added to the queue, the QEND pointer is adjusted as indiated by the NEXT pointer. This inserts the buffer in its proper location in the queue. After 22 buffers are used, the values in QSTART and QEND are both adjusted to allow for the circulation of the valid data. This procedure, which sets at most two pointers, speeds the operation and insures it will be accomplished before the receipt of the acknowledgment packet.

4.6 ALLOC Subroutine

The ALLOC routine is modified to allow the implementation of the circular buffer. If the end of the buffer is ever indicated by the value in STKPTR going to zero, the WRPFLG indicator is set and a new allocation of 16 buffers is set aside by resetting STKPTR to BUFCNT-1. Note that WRPFLG is incremented each time this occurs. If WRPFLG ever loops back to zero, it will automatically be reset to one to insure an accurate indication to the OUTDEV subroutine.

4.7 ENQ Subroutine

The ENQ routine is modified in a like manner to allow the operation of the circular buffer. It uses the QSTART and QEND pointers to control the positioning of data buffers in the queue. On the first pass through the buffer stack only QEND is adjusted to add the currently processed buffer to the queue. After 21 buffers are processed, WRPFLG is set and QSTART is also set to insure the proper circulation of the queue.

4.8 PTSTR Subroutine

As was mentioned above, the PTSTR routine is modified to allow for the transmission of a string of "null" characters before each message to the terminal device. This modification involves the addition of a preface routine which calls the old PUTSTR subroutine. This preface code saves the original message pointer and

substitutes a pointer to the null string. Once the null string is printed, the old message pointer is retrieved and the message transmitted normally.

APPENDIX I

BUS LISTENER SOFTWARE LISTING

```
CARD #  LOC   CODE    CARD  10        20        30        40        50        60        70
     2  0000            .OPT NOCNT,XREF,MEM,LIST,ERR,GEN
     3  0000          ;
     4  0000          ;THIS IS THE NEW NASA BUS LISTENER BIU CODE AS OF 06/27/79.
     5  0000          ;
     6  0000
     7  0000          ; DEVICE ADDRESSES FOLLOW
     8  0000          ;
     9  0000          NUARTS   = $C00          NETWORK UART STATUS
    10  0000          NUARTD   = $C01          NETWORK UART DATA
    11  0000          DUARTS   = $1400         DEVICE UART STATUS
    12  0000          DUARTD   = $1401         DEVICE UART DATA
    13  0000          ;
    14  0000          ; THE NEXT EQUATES REFLECT THE ACTUAL OPERATIONAL PARAMETERS
    15  0000          ; OF THE BIU WITH RESPECT TO BUFFER COUNT AND SIZE
    16  0000          ;
    17  0000          RAMSIZ   = 3072          NUMBER OF BYTES OF RAM AVAILABLE
    18  0000          BUFLEN   = 128           NUMBER OF BYTES IN A BUFFER (DO NOT CHANGE)
    19  0000          BUFMEM   = RAMSIZ-256    AMOUNT OF MEMORY ALLOCATED TO BUFFERS
    20  0000          BUFCNT   = BUFMEM/BUFLEN NUMBER OF BUFFERS AVAILABLE
```

```
CARD #  LOC   CODE    CARD  10        20        30        40        50        60        70
   22   0000
   23   0000
   24   0000
   25   0000  00
   26   0001  00 00
   27   0003  00
   28   0004  00 00
   29   0006  00
   30   0007  00
   31   0008  00
   32   0009  00
   33   000A  00
   34   000B  00
   35   000C  00
   36   000D  00
   37   000E  00
   38   000F  00
   39   0010  00
   40   0011  00
   41   0012  00
   42   0013  00
   43   0014  00
   44   0015  00
   45   0016  00
   46   0017  00
   47   0018  00
   48   0019
   49   001C
   50   001C
   51   001C
   52   001C  00
   53   001D
   54   001D
   55   0033
   56   0033
   57   0033
   58   0033  00
   59   0034  00
   60   0035
   61   0035
   62   0035
   63   0035
   64   0048
   65   0061
   66   0061

;
; THE FOLLOWING VARIABLES ARE CLEARED WHENEVER RESET IS HIT.
;
INTBC     .BYTE  0        BYTE COUNT USED BY NINT
INTPTR    .DBYTE 0        BUFFER PTR USED BY INDEV
INBC      .BYTE  0        BYTE COUNT USED BY INDEV
OUTPTR    .DBYTE 0        BUFFER PTR USED BY OUTDEV
OUTBC     .BYTE  0        BYTE COUNT USED BY OUTDEV
OUTPL     .BYTE  0        LENGTH OF PACKET BEING SENT TO DEVICE
CURNET    .BYTE  0        THE CURRENT PACKET BEING READ IN FROM THE NET
OUTSET    .BYTE  0        WHETHER A DEVICE OUTPUT BUFFER IS SET UP
INTSET    .BYTE  0        WHETHER A NETWORK INPUT BUFFER IS SET UP
FLEET     .BYTE  0        FREE RUNNING FLAG
LIVCNT    .BYTE  0        LIVE COUNTER FOR OUTPUT FORMAT
SIXPTR    .BYTE  0        INDEX OF TOP OF FREE BUFFER STACK
HOME1     .BYTE  0        ADDRESS OF FIRST MONITOR NODE
HOME2     .BYTE  0        ADDRESS OF SECOND MONITOR NODE
ALFLG     .BYTE  0        ALL ADDRESS FLAG
INCNTR    .BYTE  0        INITIAL MESSAGE COUNTER
MSGCNT    .BYTE  0        MONITOR
SKPCNT    .BYTE  0        SKIP MESSAGE COUNTER
TEMP      .BYTE  0        FIRST TEMP AREA
TEMP0     .BYTE  0        SECOND TEMP AREA
TEMP1     .BYTE  0        THIRD TEMP AREA
ECHO      .BYTE  0        ECHO BACK FLAG
WRPFLG    .BYTE  0        BUFFER MEMORY WRAP AROUND FLAG
STRING    *=*+3           INPUT STORAGE FOR INITIAL RESPONSES
;
; THE BUFFERS EACH HAVE AN ENTRY IN THIS ARRAY
;
CONECT    .BYTE 0
;
NEXT      *=*+BUFCNT      THE NEXT POINTER FOR EACH BUFFER
;
; THE QUEUE POINTERS FOLLOW
;
QSTART    .BYTE  0        THE STARTING POINTER FOR THE QUEUE OF BUFFERS
QEND      .BYTE  0        THE ENDING POINTER FOR THE QUEUE OF BUFFERS
;
; THESE ARE THE POINTERS TO THE BUFFERS
;
LOPTR     *=*+BUFCNT      THE LOW HALF OF THE PTRS
HIPTR     *=*+BUFCNT      THE HIGH HALF OF THE PTRS
;
BUFSTK    *=*+BUFCNT      THE FREE BUFFER STACK
```

```
CARD # LOC  CODE       CARD 10        20        30        40        50        60        70
   68 0077           ;
   69 0077           ; THE CODE FOLLOWS. EXECUTION COMES HERE WHEN RESET IS HIT.
   70 0077           ;
   71 0077                 *=$F800
   72 F800 D8        RESET CLD              DON'T WANT DECIMAL MODE
   73 F801 A2 7F           LDX #$7F         INITIALIZE THE STACK POINTER
   74 F803 9A              TXS
   75 F804           ;
   76 F804 A2 5B           LDX #%01011011   RESET NETWORK UART
   77 F806 8E 00 0C        STX NUARTS
   78 F809 A2 D8           LDX #%11011000   INITIALIZE NETWORK UART
   79 F80B 8E 00 0C        STX NUARTS
   80 F80E A2 57           LDX #%01010111   RESET DEVICE UART
   81 F810 8E 00 14        STX DUARTS
   82 F813 A2 16           LDX #%00010110   INITIALIZE DEVICE JART
   83 F815 8E 00 14        STX DUARTS
   84 F818           ;
   85 F818 A2 1F           LDX #CONVECT-1   ZERO OUT FIRST 1F LOCATIONS
   86 F81A A9 00           LDA #00
   87 F81C 95 00     R0    STA 0,X
   88 F81E CA              DEX
   89 F81F 10 FB           BPL R0
   90 F821           ;
   91 F821 A2 16           LDX #BUFCNT          INITIALLY, BUFCNT ITEMS IN THE STACK
   92 F823 86 0D           STX STKPTR
   93 F825 CA              DEX                  SET THE QUEUE POINTERS TO NULL
   94 F826 86 33           STX QSTART
   95 F828 86 34           STX QEND
   96 F82A           ;
   97 F82A           ; A STILL SET TO ZERO FROM ABOVE LOOP
   98 F82A           ;
   99 F82A 95 48           STA HIPTR,X          INITIALIZE FIRST BUFFER TO $0080
  100 F82C A9 80           LDA #<$0080
  101 F82E 95 35           STA LOPTR,X
  102 F830           ;
  103 F830           ;          INITIALIZE THE FREE BUFFER STACK AND THE NEXT LIST
  104 F830 A0 14           LDY #BUFCNT-2
  105 F832 8A       STACKI TXA
  106 F833 95 51           STA BUFSTK,X    PUT THE NUMBER OF EACH BUFFER IN THE STACK
  107 F835 94 1D           STY NEXT,X
  108 F837 88              DEY
  109 F838 CA              DEX
  110 F839 10 F7           BPL STACKI
  111 F83B E8              INX             SET LAST NEXT POINTER
  112 F83C A0 15           LDY #BUFCNT-1
  113 F83E 94 1D           STY NEXT,X
  114 F840           ;
  115 F840 A2 14           LDX #BUFCNT-2
  116 F842 A9 80           LDA #<$0180     SET THE REMAINING BUFFER LOW AND HI POINTERS...
  117 F844 A0 01           LDY #>$0180       ...STARTING AT ADDRESS $0180
  118 F846 95 35    BUFFRI STA LOPTR,X     SET THE LOW HALF ADDRESSES
  119 F848 98              TYA
  120 F849 95 48           STA HIPTR,X     SET THE HIGH HALF
  121 F84B B5 35           LDA LOPTR,X     GET BACK THE LOW HALF FOR INCREMENTING
  122 F84D 18              CLC
```

```
CARD #  LOC   CODE       CARD  10        20        30        40        50        60        70
  123   F84E  69 80            ADC #BUFLEN       ADD IN ONE BUFFERS LENGTH
  124   F850  90 01            BCC SKIPI         IF NO CARRY DON'T INCR HIGH HALF
  125   F852  C8               INY               IF CARRY THEN HIGH HALF WILL BE INCREMENTED
  126   F853  CA        SKIPI  DEX
  127   F854  10 F0            BPL BUFFRI        CONTINUE UNTIL ALL BUFFER ADDR'S SET
  128   F856                  ;
  129   F856  A0 1C            LDY #CRLF         PRINT CR/LF
  130   F858  20 73 FB         JSR PUTSTR
  131   F85B  A5 1C            LDA CONECT        GET OLD CONNECT FLAG
  132   F85D  A2 FF            LDX #$FF          AND SET CONECT TO -1
  133   F85F  86 1C            STX CONECT
  134   F861  D0 05            BNE I1            WERE WE LISTENING BEFORE?
  135   F863  A0 95            LDY #PROINT       YES, SO PRINT "PROCESSING INTERRUPTED"
  136   F865  20 68 FB         JSR PTSTR
  137   F868                  ;
  138   F868  A0 00     I1     LDY #PKCNT        PRINT PACKET COUNT REQUEST
  139   F86A  20 68 FB         JSR PTSTR
  140   F86D  20 9C F8         JSR INDEV         GET RESPONSE
  141   F870  A0 1F            LDY #PKSKP        PRINT PACKET SKIP REQUEST
  142   F872  20 68 FB         JSR PTSTR
  143   F875  20 9C F8         JSR INDEV         GET RESPONSE
  144   F878  A0 41            LDY #MNADD        PRINT MONITOR ADDRESS REQUEST
  145   F87A  20 68 FB         JSR PTSTR
  146   F87D  20 9C F8         JSR INDEV         GET RESPONSE
  147   F880  A4 10            LDY ALFLG         ARE WE MONITORING ALL ADDRESSES
  148   F882  D0 08            BNE I2            YES, SO SKIP NEXT REQUEST
  149   F884  A0 5A            LDY #SCADD        NO, PRINT SECOND ADDRESS REQUEST
  150   F886  20 68 FB         JSR PTSTR
  151   F889  20 9C F8         JSR INDEV         GET RESPONSE
  152   F88C  20 8E FA  I2     JSR INTBUF        SET UP A NETWORK INPUT BUFFER
  153   F88F  58               CLI               ALLOW INTERRUPTS
  154   F890                  ;
  155   F890                  ; MLOOP IS THE MAIN LOOP. IT REPEATEDLY CALLS INDEV, OUTDEV AND
  156   F890                  ; INTBUF UNTIL RESET IS HIT BY THE USER.
  157   F890                  ;
  158   F890  20 C9 F9  MLOOP  JSR OUTDEV        POLL FOR DEVICE OUTPUT
  159   F893  20 8E FA         JSR INTBUF        SEE IF A NETWORK INPUT BUFFER IS NECESSARY
  160   F896  20 9C F8         JSR INDEV         POLL FOR DEVICE INPUT
  161   F899  4C 90 F8         JMP MLOOP         AND LOOP FOREVER
```

```
CARD #  LOC   CODE        CARD  10        20        30        40        50        60        70
  163   F89C              ;
  164   F89C              ; INDEV IS CALLED TO POLL FOR CHARACTERS FROM THE DEVICE. IT IS USED
  165   F89C              ; WHEN THE DEVICE IS ENTERING PACKETS TO CONTROL THE BIU.
  166   F89C              ; IT TRIES TO GET A CHAR. IF IT IS SUCCESSFUL, THE CHARACTER IS
  167   F89C              ; ADDED TO THE BUFFER.
  168   F89C              ;
  169   F89C AD 00 14     INDEV   LDA DUARTS      IS THERE ANY DATA READY FROM THE DEVICE?
  170   F89F 29 01                AND #%00000001
  171   F8A1 D0 05                BNE GOO         YES, CONTINUE
  172   F8A3 A5 1C                LDA CONECT      ARE WE WAITING FOR A RESPONSE?
  173   F8A5 30 F5                BMI INDEV       YES, SO TRY AGAIN
  174   F8A7 60           IRET    RTS             NO, SO RETURN
  175   F8A8 AD 01 14     GOO     LDA DUARTD      GET THE DATA
  176   F8AB 29 7F                AND #$7F        GET RID OF THE PARITY
  177   F8AD AA                   TAX             PUT THE DATA IN X
  178   F8AE A5 17                LDA ECHO        SHOULD WE ECHO TO THE USER?
  179   F8B0 D0 0A                BNE NOECHO      NO
  180   F8B2 A9 02                LDA #%00000010
  181   F8B4 2C 00 14     IN0     BIT DUARTS      IS IT OK TO OUTPUT TO THE TERMINAL?
  182   F8B7 F0 FB                BEQ IN0         NOT YET
  183   F8B9 8E 01 14             STX DUARTD      ECHO THE CHAR
  184   F8BC 8A           NOECHO  TXA             PUT THE CHAR IN A
  185   F8BD A6 1C                LDX CONECT      GET THE CURRENT STATE OF THE DEVICE
  186   F8BF 30 13                BMI GETRPY      IF WAITING FOR REPLY TO INITIAL QUESTION
  187   F8C1 C9 1B                CMP #$1B        LOOKING FOR A ESC
  188   F8C3 F0 02                BEQ GO1         YES, SO CONTINUE
  189   F8C5 D0 E0                BNE IRET        NO, SO RETURN
  190   F8C7 A9 00        GO1     LDA #00         YES, SO ZERO OUT SKIP COUNT
  191   F8C9 85 13                STA SKPCNT      AND MESSAGE COUNT
  192   F8CB 85 12                STA MSGCNT      SEND PROCESSING INTERRUPTED MESSAGE
  193   F8CD A0 95                LDY #PROINT
  194   F8CF 20 68 FB             JSR PTSTR
  195   F8D2 78                   SEI             DON'T WANT INTERUPTS NOW
  196   F8D3 60                   RTS             RETURN
  197   F8D4              ;
  198   F8D4              ; GETRPY ACCUMULATES THE RESPONSES TO THE INITIAL SIGN-ON QUESTIONS
  199   F8D4              ;
  200   F8D4 85 14        GETRPY  STA TEMP        GET RID OF LOWER CASE
  201   F8D6 A9 40                LDA #$40
  202   F8D8 24 14                BIT TEMP        IS THIS AN ALPHA CHARACTER?
  203   F8DA F0 06                BEQ IN1         NO
  204   F8DC A5 14                LDA TEMP        YES, SO MAKE SURE IT'S UPPER CASE
  205   F8DE 29 5F                AND #$5F
  206   F8E0 D0 02                BNE IN2         ALWAYS BRANCH
  207   F8E2 A5 14        IN1     LDA TEMP
  208   F8E4 A6 03        IN2     LDX INBC        THE OFFSET IN STRING TO PUT THE CHAR
  209   F8E6 95 19                STA STRING,X
  210   F8E8 C9 0D                CMP #$0D        WAS THE CHAR A CR?
  211   F8EA F0 08                BEQ GLF         YES, SO OUTPUT A LF
  212   F8EC E0 02                CPX #$02        WAS IT THE THIRD CHAR?
  213   F8EE F0 0E                BEQ GCRLF       YES, SO OUTPUT A CRLF
  214   F8F0 E6 03                INC INBC        NEITHER OF THE ABOVE, SO WAIT FOR MORE INPUT
  215   F8F2 D0 A8                BNE INDEV       ALWAYS BRANCH
  216   F8F4              ;
  217   F8F4 C6 03        GLF     DEC INBC        SUBTRACT ONE FROM INBC FOR THE CR
```

```
CARD #  LOC   CODE        CARD      10        20        30        40        50        60        70
  218   F8F6  10 02               BPL  GLF1        WAS ONLY THE CR ENTERED?
  219   F8F8  E6 05               INC  INBC        YES, SO PROCESS IT AS THE INPUT
  220   F8FA  A0 1D         GLF1  LDY  #LF
  221   F8FC  D0 02               BNE  GPUT        ALWAYS BRANCH
  222   F8FE  A0 1C         GCRLF LDY  #CRLF
  223   F900  20 73 FB      GPUT  JSR  PUTSTR      PRINT THE CHARS
  224   F903                      ;
  225   F903  A2 00               LDX  #00         INSURE THAT RESPONSE IS LEGAL
  226   F905  A0 00         G0    LDY  #00
  227   F907  B5 19         G1    LDA  STRING,X
  228   F909  D9 54 FC            CMP  TABLE,Y
  229   F90C  D0 07               BNE  NMATCH      NOT A NUMERAL
  230   F90E  E4 05               CPX  INBC        IS IT THE LAST CHARACTER
  231   F910  F0 37               BEQ  MATCH       YES
  232   F912  E8                 INX              NO, SO GET THE NEXT CHARACTER
  233   F913  D0 F0               BNE  G0          ALWAYS BRANCH
  234   F915                      ;
  235   F915  C8            NMATCH INY              CHECK THE NEXT NUMBER
  236   F916  C0 10               CPY  #16         HAVE WE LOOKED AT ALL 16 POSSIBILITIES
  237   F918  30 EB               BMI  G1          NO, SO TRY AGAIN
  238   F91A  A2 00               LDX  #00         YES, SO SEE IF IT IS THE WORD "ALL"
  239   F91C  A0 00               LDY  #00
  240   F91E  B5 19         G2    LDA  STRING,X
  241   F920  D9 57 FC            CMP  ALL,Y
  242   F923  D0 08               BNE  ASKAGN      NOT "ALL", SO ERROR-RESTART
  243   F925  E0 02               CPX  #02         THE THIRD LETTER YET?
  244   F927  F0 48               BEQ  MATCHA      YES
  245   F929  E8                 INX              NO, GET NEXT LETTER
  246   F92A  C8                 INY
  247   F92B  D0 F1               BNE  G2          ALWAYS BRANCH
  248   F92D                      ;
  249   F92D  C9 00         ASKAGN CMP  #$00        WAS THIS CHARACTER A CR
  250   F92F  D0 10               BNE  AK2         NO, SO ASK AGAIN
  251   F931  A5 0E               LDA  HOME1       SET UP FOR ONE ADDRESS ONLY
  252   F933  A4 11               LDY  INCNTR      WAS THIS THE RESPONSE TO THE SECOND ADDRESS?
  253   F935  C0 03               CPY  #$03
  254   F937  F0 10               BEQ  MATCH       YES, SO GO TO MATCH
  255   F939  C0 02               CPY  #$02        WAS THIS THE 1ST OR 2ND RESPONSE
  256   F93B  F0 04               BEQ  AK1         NO, SO ERROR INPUT
  257   F93D  A9 00               LDA  #00         YES, SO ASSUME A ZERO INPUT.
  258   F93F  F0 08               BEQ  MATCH
  259   F941  A0 78         AK1   LDY  #ERR        PRINT ERROR MESSAGE
  260   F943  20 68 FB            JSR  PTSTR
  261   F946  4C 00 F8            JMP  RESET       START AGAIN
  262   F949                      ;
  263   F949  E6 11         MATCH INC  INCNTR      COUNT THE CORRECT RESPONSES
  264   F94B  20 97 F9            JSR  CTON        CONVERT INPUT TO A HEX NUMBER
  265   F94E  A0 00               LDY  #00         PREPARE TO SET COUNTERS IF NECESSARY
  266   F950  A5 11               LDA  INCNTR
  267   F952  C9 02               CMP  #02
  268   F954  30 0B               BMI  M0          THE FIRST CORRECT RESPONSE
  269   F956  F0 14               BEQ  M1          SECOND CORRECT RESPONSE
  270   F958  C9 04               CMP  #04
  271   F95A  30 14               BMI  M2          THIRD     "       "
  272   F95C  86 0F               STX  HOME2       SAVE THE SECOND LISTENING ADDRESS
```

```
CARD #  LOC   CODE       CARD  10        20        30        40        50        60        70
  273   F95E  4C 7C F9          JMP READY       START LISTENING
  274   F961                ;
  275   F961  E0 00       M0    CPX #00         DO WE WANT FREE RUNNING MODE?
  276   F963  D0 03             BNE M0R         NO, SO STORE X
  277   F965  E8                INX             YES, SO SET FLAGS
  278   F966  86 09             STX FLEET
  279   F968  86 12       M0R   STX MSGCNT      SAVE THE MESSAGE COUNT
  280   F96A  D0 14             BNE MRET        ALWAYS BRANCH
  281   F96C                ;
  282   F96C  86 15       M1    STX SKPCNT      SAVE THE SKIP COUNT
  283   F96E  F0 10             BEQ MRET        AND RETURN
  284   F970                ;
  285   F970  86 0E       M2    STX MONE1       SAVE THE FIRST MONITOR ADDRESS
  286   F972  30 0C             BMI MRET        AND RETURN
  287   F974                ;
  288   F974  A5 11       MATCHA LDA INCNTR     IS THIS A RESPONSE TO THE THIRD OR FOURTH
  289   F976  C9 02              CMP #02           REQUEST?
  290   F978  30 03              BMI ASKAGN      NO, SO ERROR-RETURN
  291   F97A  E6 10              INC ALFLG       YES, SO SET ALL FLAG
  292   F97C  E6 1C       READY  INC CONECT      SET LISTENING FLAG
  293   F97E  84 11              STY INCNTR      RESET COUNTERS
  294   F980  84 03       MRET   STY INBC
  295   F982  60                 RTS             AND RETURN
  296   F983                ;
  297   F983  85 15       ALPHA  STA TEMP0       SAVE THE INPUT CHARACTER
  298   F985  A9 40              LDA #$40        IS THE CHARACTER A NUMERAL?
  299   F987  24 15              BIT TEMP0
  300   F989  F0 07              BEQ A1          YES
  301   F98B  A5 15              LDA TEMP0       NO, SO ADD NINE TO IT
  302   F98D  18                 CLC
  303   F98E  69 09              ADC #$09
  304   F990  D0 02              BNE A2          ALWAYS BRANCH
  305   F992  A5 15       A1    LDA TEMP0
  306   F994  29 0F       A2    AND #$0F
  307   F996  60                 RTS             RETURN
  308   F997                ;
  309   F997  A6 03       CTON   LDX INBC        CONVERT CHARACTER TO NUMERAL (HEX)
  310   F999  F0 19              BEQ C1          ONLY A CARRIAGE RETURN, SO SET DUMMY IN X.
  311   F99B  E0 02              CPX #02         WERE THERE THREE CHARACTERS INPUT?
  312   F99D  10 17              BPL C2          YES
  313   F99F  B5 19       C0     LDA STRING,X    NO, SO GET LOW ORDER CHARACTER
  314   F9A1  20 83 F9           JSR ALPHA       CONVERT TO BINARY HALFWORD
  315   F9A4  85 16              STA TEMP1
  316   F9A6  CA                 DEX
  317   F9A7  30 08              BMI C1          LAST CHAR?
  318   F9A9  B5 19              LDA STRING,X    NO, SO GET HIGH ORDER CHARACTER
  319   F9AB  20 83 F9           JSR ALPHA       CONVERT TO BINARY HALFWORD
  320   F9AE  0A                 ASL A           AND SHIFT IT INTO HIGH ORDER SPOT
  321   F9AF  0A                 ASL A
  322   F9B0  0A                 ASL A
  323   F9B1  0A                 ASL A
  324   F9B2  05 16              ORA TEMP1       COMBINE HALVES
  325   F9B4  AA          C1     TAX             PUT IT IN X REG
  326   F9B5  60          CRTN   RTS             AND RETURN
  327   F9B6                ;
```

```
CARD #  LOC   CODE        CARD  10        20        30        40        50        60        70
  328   F9B6  A2 00       C2          LDX #00          CHECK FIRST CHARACTER ENTERED,
  329   F9B8  B5 19                   LDA STRING,X     IT MUST BE EITHER 0 OR A BLANK.
  330   F9BA  C9 30                   CMP #$30         IS IT ZERO?
  331   F9BC  F0 07                   BEQ C3           YES
  332   F9BE  C9 20                   CMP #$20         IS IT A BLANK?
  333   F9C0  F0 03                   BEQ C3           YES
  334   F9C2  4C 20 F9                JMP ASKAGN       NEITHER, SO ERROR-RESET
  335   F9C5  A6 03       C3          LDX INBC
  336   F9C7  D0 D6                   BNE C0           ALWAYS BRANCH
  337   F9C9              ;
```

```
CARD #  LOC   CODE        CARD  10        20        30        40        50        60        70
  339   F9C9              ;
  340   F9C9              ; OUTDEV IS CALLED TO HANDLE A MESSAGE FROM THE NETWORK. IT WILL
  341   F9C9              ; TRANSMIT THE MESSAGE TO THE USER DEVICE.
  342   F9C9              ;
  343   F9C9  A5 12       OUTDEV  LDA MSGCNT      HAS THE MSG COUNT LAPSED?
  344   F9CB  F0 01               BEQ OT1         YES, SO CONTINUE
  345   F9CD  60                  RTS             NO, SO KEEP SAVING MESSAGES
  346   F9CE  78            OT1   SEI             HALT INTERRUPTS
  347   F9CF  A4 18               LDY WRPFLG      CHECK TO SEE IF OUTPUT BUFFER WRAPPED AROUND
  348   F9D1  F0 03               BEQ OT2         NO, SO CONTINUE.
  349   F9D3  A6 34               LDX QEND        YES, SO PREFORM SHIFT IN QEND AND QSTART
  350   F9D5  86 33               STX QSTART      THIS IS NECESSARY TO OUTPUT DATA IN
  351   F9D7  E8                  INX                   PROPER ORDER.
  352   F9D8  E0 16               CPX #BUFCNT
  353   F9DA  D0 02               BNE OT01
  354   F9DC  A2 00               LDX #00
  355   F9DE  86 34         OT01  STX QEND
  356   F9E0  A5 09         OT2   LDA OUTSET      SET UP FOR OUTPUT?
  357   F9E2  D0 29               BNE PUTCH1      YES, SO SEE IF CAN SEND THE NEXT CHARACTER
  358   F9E4  A6 33               LDX QSTART      ANYTHING ON THE QUEUE?
  359   F9E6  E4 34               CPX QEND
  360   F9E8  D0 0F               BNE OT22        YES, SO CONTINUE
  361   F9EA  A4 18               LDY WRPFLG      CHECK TO SEE IF OUTPUT BUFFER WRAPPED
  362   F9EC  D0 03               BNE OT21        YES, SO ALLOW ONE MORE OUTPUT.
  363   F9EE  4C 7A FA            JMP OUTRET      NO, SO EXIT
  364   F9F1  A0 00         OT21  LDY #00         ZERO OUT WRAP FLAG
  365   F9F3  84 18               STY WRPFLG
  366   F9F5  B5 19               LDA NEXT,X
  367   F9F7  85 34               STA QEND        SET SO QSTART = QEND AFTER THIS CYCLE
  368   F9F9  B4 35         OT22  LDY LOPTR,X     SET UP THE PTRS
  369   F9FB  84 04               STY OUTPTR
  370   F9FD  B4 44               LDY HIPTR,X
  371   F9FF  84 05               STY OUTPTR+1
  372   FA01  E6 09               INC OUTSET      SET UP FOR OUTPUT
  373   FA03  A0 FF               LDY #$FF
  374   FA05  84 06               STY OUTBC       FIRST CHAR OFFSET - 1
  375   FA07  A0 07               LDY #07
  376   FA09  B1 04               LDA (OUTPTR),Y  GET THE PACKET LENGTH
  377   FA0B  85 07               STA OUTPL
  378   FA0D              ;
  379   FA0D              ; PUTCH TRIES TO PUT A CHARACTER FROM THE BUFFER TO THE COMPUTER.
  380   FA0D              ;
  381   FA0D  A0 C6       PUTCH1  LDY #NULL       PRINT 6 NULL CHARACTERS ON OUTPUT DEVICE
  382   FA0F  20 73 FB            JSR PUTSTR
  383   FA12  A0 00               LDY #00
  384   FA14  84 0C               STY LINCNT
  385   FA16  A4 06       PUTCH   LDY OUTBC       GET THE NEXT CHAR OFFSET
  386   FA18  C4 07               CPY OUTPL       DONE YET?
  387   FA1A  D0 05               BNE TRYDS       NO, SO CHECK THE DEVICE STATUS
  388   FA1C  C6 09               DEC OUTSET      YES, SO ALL DONE
  389   FA1E  4C 5A FA            JMP OUTFRE
  390   FA21  AD 00 14    TRYDS   LDA OUARTS      WILL DEVICE TAKE A CHARACTER?
  391   FA24  29 02               AND #%00000010
  392   FA26  F0 F9               BEQ TRYDS       NO, SO KEEP WAITING
  393   FA28  C8                  INY             GET THE OFFSET OF THE CHAR TO SEND
```

```
CARD #  LOC   CODE        CARD   10        20        30        40        50        60        70
  394   FA29  B1 04              LDA (OUTPTR),Y  GET THE CHAR TO SEND
  395   FA2B  84 06              STY OUTBC       SAVE THE POINTER TO THE CHARACTER
  396   FA2D  29 F0              AND #$F0        MODIFY IT FOR HEX OUTPUT
  397   FA2F  18                 CLC
  398   FA30  6A                 ROR A           ROTATE IT FOUR TIMES
  399   FA31  6A                 ROR A
  400   FA32  6A                 ROR A
  401   FA33  6A                 ROR A
  402   FA34  A8                 TAY
  403   FA35  B9 5A FC           LDA TABLE,Y     GET THE HEX VALUE FROM THE TABLE
  404   FA38  8D 01 14           STA DUARTD      OUTPUT THE HEX CHARACTER
  405   FA3B  AD 00 14   TO      LDA DUARTS      WILL THE DEVICE TAKE THE CHARACTER?
  406   FA3E  29 02              AND #$02
  407   FA40  F0 F9              BEQ TO          NO, SO TRY AGAIN
  408   FA42  A4 06              LDY OUTBC       RECALL THE CHARACTER POINTER
  409   FA44  B1 04              LDA (OUTPTR),Y  YES, SO LOAD CHARACTERS FOR SECOND HALF
  410   FA46  29 0F              AND #$0F        GET RID OF HIGH ORDER BYTE
  411   FA48  A8                 TAY
  412   FA49  B9 5A FC           LDA TABLE,Y     LOAD THE HEX CHARACTER FROM THE TABLE
  413   FA4C  8D 01 14           STA DUARTD      OUTPUT THE CHARACTER
  414   FA4F  A0 CC              LDY #BLANK      OUTPUT A BLANK SPACE
  415   FA51  20 73 FB           JSR PUTSTR
  416   FA54  E6 0C              INC LINCNT      COUNT THE CHARACTERS ON A LINE
  417   FA56  A4 0C              LDY LINCNT      HAVE WE PRINTED A LINE FULL YET?
  418   FA58  C0 10              CPY #16
  419   FA5A  30 BA              BMI PUTCH       NO, SO GET THE NEXT CHARACTER
  420   FA5C  A0 00              LDY #00         YES, SO RESET THE LINE COUNTER
  421   FA5E  84 0C              STY LINCNT
  422   FA60  A0 1C              LDY #CRLF       AND PRINT A CR LF
  423   FA62  20 73 FB           JSR PUTSTR
  424   FA65  4C 0D FA           JMP PUTCH1      AND THE NEW LINE NULLS
  425   FA68                 ;
  426   FA68  A2 16      OUTFRE  LDX #OUSTART-NEXT FREE UP THE PACKET
  427   FA6A  20 59 FB           JSR OU
  428   FA6D  A0 1C              LDY #CRLF       OUTPUT A CR LF BETWEEN MESSAGES
  429   FA6F  20 73 FB           JSR PUTSTR
  430   FA72  A0 1C              LDY #CRLF
  431   FA74  20 73 FB           JSR PUTSTR      AGAIN
  432   FA77  4C E0 F9           JMP OT2         AND GO GET THE NEXT MESSAGE
  433   FA7A                 ;
  434   FA7A  A4 12      OUTRET  LDY MSGCNT
  435   FA7C  30 05              BMI NORMAL      NORMAL TERMINATION
  436   FA7E  F0 04              BEQ NORMAL
  437   FA80  A0 E1              LDY #ENDERR     ABNORMAL TERMINATION
  438   FA82  D0 04              BNE SNDMES      (ALWAYS BRANCH)
  439   FA84  A0 AF      NORMAL  LDY #FINISH     SEND THE PROCESSING FINISHED MESSAGE
  440   FA86  C6 1C              DEC CONECT      NO LONGER QUEUING PACKETS
  441   FA88  20 68 FB   SNDMES  JSR PTSTR       SEND THE MESSAGE
  442   FA8B  4C 00 F8           JMP RESET       AND START OVER AGAIN
```

```
CARD #  LOC   CODE       CARD  10        20        30        40        50        60        70
  444   FA8E             ;
  445   FA8E             ; INTBUF SETS UP A BUFFER FOR NETWORK INPUT IF IT IS NECESSARY.
  446   FA8E             ;
  447   FA8E  A5 04      INTBUF   LDA INTSET      IS A BUFFER SET UP FOR NETWORK INPUT?
  448   FA90  D0 14              BNE INTRET       YES
  449   FA92  08                 PHP              SAVE THE INTERRUPT STATUS
  450   FA93  78                 SEI              TURN OFF INTERRUPTS, IF NOT ALREADY OFF
  451   FA94  20 34 FB           JSR ALLOC        ALLOCATE A BUFFER
  452   FA97  30 0E              BMI INTRES       IF CAN'T
  453   FA99  28                 PLP              RESTORE OLD INTERRUPT STATUS
  454   FA9A  B5 55              LDA LOPTR,X      SET UP THE PTR TO THE BUFFER
  455   FA9C  85 01              STA INTPTR
  456   FA9E  B5 48              LDA HIPTR,X
  457   FAA0  85 02              STA INTPTR+1
  458   FAA2  86 04              STX CURNET       SAVE THE BUFFER NUMBER
  459   FAA4  E6 04              INC INTSET       NOW ALL SET UP
  460   FAA6  60         INTRET   RTS
  461   FAA7  28         INTRES   PLP              RESTORE OLD INTERRUPT STATUS AND RETURN
  462   FAA8  60                  RTS
```

```
CARD #  LOC   CODE        CARD  10        20        30        40        50        60        70
  464   FAA9              ;
  465   FAA9              ; IRQ RECEIVES CONTROL WHENEVER THERE IS AN INTERRUPT
  466   FAA9              ;
  467   FAA9  48          IRQ       PHA             PUSH ACC AND Y ONTO STACK
  468   FAAA  98                    TYA
  469   FAAB  48                    PHA
  470   FAAC              ;
  471   FAAC              ; NINT IS THE NETWORK UART INTERRUPT HANDLER. IT IS USED TO EITHER
  472   FAAC              ; CHECK LAT VALIDITY, OR TO READ IN A MSG FROM ANOTHER BIU.
  473   FAAC              ;
  474   FAAC  AD 00 0C    NINT      LDA NUARTS      GET THE STATUS
  475   FAAF  AC 01 0C              LDY NUARTD      AND THE DATA
  476   FAB2  85 00                 STA INTBC       SAVE THE PARITY ERROR FLAG
  477   FAB4              ;
  478   FAB4  8A                    TXA             SAVE X
  479   FAB5  48                    PHA
  480   FAB6  98                    TYA             GET THE TRANSMITTED ADDRESS
  481   FAB7  A6 10                 LDX ALFLG       IS THE ALL ADDRESS.FLAG SET?
  482   FAB9  E0 01                 CPX #01
  483   FABB  F0 08                 BEQ N13         YES, SO SKIP THE ADDRESS CHECK
  484   FABD  C5 0E                 CMP HOME1       IS THIS ONE OF THE MONITORED DEVICES?
  485   FABF  F0 04                 BEQ N13         YES
  486   FAC1  C5 0F                 CMP HOME2       IS IT THE OTHER DEVICE?
  487   FAC3  D0 D3                 BNE N5          NO
  488   FAC5  A4 13       N13       LDY SKPCNT      HAVE WE SKIPPED ENOUGH MESSAGES YET?
  489   FAC7  F0 05                 BEQ N3          YES
  490   FAC9  C6 13                 DEC SKPCNT      NO, SO SUBTRACT ONE MORE
  491   FACB  4C D2 FA              JMP N5          ALWAYS BRANCH
  492   FACE  A4 0A       N3        LDY INTSET      ARE WE ALL SET UP FOR INPUT?
  493   FAD0  D0 0A                 BNE N6          YES
  494   FAD2  A9 B8       N5        LDA #%01011000  CAN'T RECEIVE, SO DISABLE RECEIVER
  495   FAD4  8D 00 0C              STA NUARTS
  496   FAD7  4C 21 FB              JMP NIRET       AND RETURN
  497   FADA              ;
  498   FADA  A0 00       N6        LDY #00         STORE BYTE 0 IN BUFFER
  499   FADC  91 01                 STA (INTPTR),Y
  500   FADE  C8          N7        INY
  501   FADF  AD 00 0C    N8        LDA NUARTS      IS ANOTHER WORD READY
  502   FAE2  29 05                 AND #%00000101  IS RECEIVE KEY ON BUT WORD NOT IN?
  503   FAE4  F0 F9                 BEQ N8          IF NOT, GO TO N8
  504   FAE6  29 04                 AND #%00000100  IS RECEIVE KEY ON?
  505   FAE8  F0 0C                 BEQ N16         IF NOT, TEST FOR AN ACK
  506   FAEA  C0 01                 CPY #01         WAS THIS AN ACK?
  507   FAEC  D0 E4                 BNE N5          NO, SO TURN OFF THE RECEIVER
  508   FAEE  A0 07                 LDY #07         YES, SO CONTINUE
  509   FAF0  A9 00                 LDA #00         SET THE MESSAGE LENGTH
  510   FAF2  91 01                 STA (INTPTR),Y
  511   FAF4  F0 17                 BEQ N10         ALWAYS BRANCH
  512   FAF6  AD 01 0C    N16       LDA NUARTD      READ THE NEXT WORD
  513   FAF9  91 01                 STA (INTPTR),Y
  514   FAFB  AA                    TAX             SAVE THE DATA CHAR
  515   FAFC  C0 07                 CPY #07         IS THIS PACKET BYTE COUNT?
  516   FAFE  90 DE                 BCC N7          IF <, KEEP ON READING IN THE HEADER
  517   FB00  D0 02                 BNE N9          IF >, COMPARE TO INTBC
  518   FB02  86 00                 STX INTBC       IF =, STORE RECEIVE PACKET LENGTH
```

```
CARD # LOC  CODE      CARD  10        20        30        40        50        60        70
  519 FB04 C4 00      N9    CPY INTBC       DOES Y = PACKET LENGTH?
  520 FB06 D0 D6            BNE N7          NO, SO KEEP ON GETTING BYTES
  521 FB08                  ;
  522 FB08 A9 58            LDA #%01011000  DISABLE THE RECEIVER
  523 FB0A 9D 03 0C         STA NUARTS
  524 FB0D A4 08      N10   LDY FLEET       ARE WE FREE RUNNING?
  525 FB0F D0 06            BNE QIT         YES, SO SKIP THE MESSAGE COUNTING
  526 FB11 A4 12            LDY MSGCNT      HAVE WE SAVED ENOUGH MESSAGES YET?
  527 FB13 F0 0C            BEQ NIRET       YES
  528 FB15 C6 12            DEC MSGCNT      NO, SO SUBTRACT ONE FROM THE COUNT
  529 FB17                  ;
  530 FB17 A6 09      QIT   LDX CURNET      QUEUE THE MESSAGE TO BE SENT IN TO THE DEVICE
  531 FB19 20 4A FB         JSR ENQ
  532 FB1C C6 0A            DEC INTSET      NO LONGER SET UP FOR INPUT
  533 FB1E 20 BE FA         JSR INTBUF      TRY TO GET SET UP AGAIN
  534 FB21                  ;
  535 FB21 68         NIRET PLA             RESTORE X
  536 FB22 AA               TAX
```

```
CARD #  LOC   CODE       CARD  10        20        30        40        50        60        70
  538   FB23             ;
  539   FB23             ; RET IS USED TO RETURN FROM ALL INTERRUPTS.
  540   FB23             ;
  541   FB23  68         RET       PLA             UNSTACK AND RETURN
  542   FB24  A8                   TAY
  543   FB25  68                   PLA
  544   FB26  40                   RTI
  545   FB27             ;
  546   FB27             ; NMI OCCURS WHEN THE RECEIVE KEY TURNS OFF.
  547   FB27             ;
  548   FB27  48         NMI       PHA             PUSH A
  549   FB28  A9 5B                LDA #%01011011  RESET NETWORK UART
  550   FB2A  8D 00 0C             STA NUARTS
  551   FB2D  A9 D8                LDA #%11011000  INITIALIZE NETWORK UART
  552   FB2F  8D 00 0C             STA NUARTS
  553   FB32  68                   PLA
  554   FB33  40                   RTI
```

```
CARD #  LOC   CODE        CARD  10        20        30        40        50        60        70
  556   FB34              ;
  557   FB34              ; ALLOC IS CALLED TO ALLOCATE A FREE BUFFER AND RETURN ITS NUMBER IN
  558   FB34              ; X. IF THERE ARE NONE LEFT, THE QUEUE WILL BE WRAPPED AROUND AND
  559   FB34              ; OVERWRITE ITSELF. X AND Y ARE CHANGED.
  560   FB34              ;
  561   FB34  A4 0D       ALLOC   LDY STKPTR        GET OFFSET OF THE TOP OF FREE BUFFER
  562   FB36  88          ALLOC1  DEY               POINT TO THE NEXT FREE BUFFER NUMBER
  563   FB37  F0 07               BEQ WRAP          START WRAP PROCESS
  564   FB39  30 04               BMI WRAP1         WRAP AROUND IF NONE AVAILABLE
  565   FB3B  84 0D       ALLOC2  STY STKPTR        GOT ONE, SO SAVE THE NEW TOP OF THE STACK
  566   FB3D  B6 61               LDX BUFSTK,Y      GET THE ALLOCATED BUFFER NUMBER
  567   FB3F  60                  RTS
  568   FB40  E6 18       WRAP    INC WRPFLG        MAKE WRAP FLAG NONZERO TO CONTROL QUEUE
  569   FB42  D0 F7               BNE ALLOC2
  570   FB44  F0 FA               BEQ WRAP
  571   FB46  A0 16       WRAP1   LDY #BUFCNT       WRAP AROUND IF CAN'T ALLOCATE A BUFFER
  572   FB48  D0 EC               BNE ALLOC1        ALWAYS BRANCH
  573   FB4A              ;
  574   FB4A              ; END IS USED TO ADD A BUFFER TO THE END OF A QUEUE.  IT USES A
  575   FB4A              ; CIRCULAR QUEUEING ROUTINE THAT HAS QSTART AS THE STARTING POINTER AND
  576   FB4A              ; QEND AS THE ENDING POINTER.  WRPFLG IS USED TO INDICATE THE THE
  577   FB4A              ; QUEUE HAS BEEN COMPLETELY FILLED AT LEAST ONCE AND THE WRAP AROUND
  578   FB4A              ; PROCESS IS AT WORK.
  579   FB4A              ;
  580   FB4A  85 1D       END     LDA NEXT,X        IS THIS THE LAST ENTRY?
  581   FB4C  85 34               STA QEND          SET THE END POINTER
  582   FB4E  A4 18               LDY WRPFLG        ARE WE WRAPING AROUN
  583   FB50  D0 01               BNE END1          YES, SO CONTINUE
  584   FB52  60                  RTS               NO, SO RETURN
  585   FB53  AA          END1    TAX               UPDATE THE STARTING POINTER
  586   FB54  B5 1D               LDA NEXT,X
  587   FB56  85 33               STA QSTART
  588   FB58  60                  RTS               AND RETURN
  589   FB59              ;
  590   FB59              ; DQ IS CALLED TO DEQUEUE AND FREE A BUFFER ON A QUEUE. X IS THE OFFSET
  591   FB59              ; FROM NEXT OF ENTRY PREVIOUS TO THE ONE TO BE FREED. X,A,Y ARE CHANGED.
  592   FB59              ;
  593   FB59  B4 1D       DQ      LDY NEXT,X        Y HAS THE NUMBER OF THE BUFFER TO BE FREED
  594   FB5B  B9 1D 00            LDA NEXT,Y        A HAS THE NUMBER OF THE NEXT BUFFER IN THE QUEUE
  595   FB5E  95 1D               STA NEXT,X        THE BUFFER IS DEQUEUED
  596   FB60  A6 0D               LDX STKPTR        ADD THE BUFFER TO THE FREE BUFFER STACK
  597   FB62  98                  TYA
  598   FB63  95 61               STA BUFSTK,X      NOW IN THE FREE BUFFER POOL
  599   FB65  E6 0D               INC STKPTR        ONE MORE FREE BUFFER
  600   FB67  60                  RTS
  601   FB68              ;
  602   FB68              ; PUTSTR IS USED TO PRINT A STRING FROM THE ASCII STRING LIST.
  603   FB68              ; IT EXPECTS THE OFFSET OF THE STRING FROM THE BEGINNING OF
  604   FB68              ; ASCII IN Y.
  605   FB68              ;
  606   FB68              ; PTSTR IS USED BEFORE THE OUTPUT OF A MESSAGE TO SOLVE A TIMING
  607   FB68              ; PROBLEM ON SOME OUTPUT DISPLAYS.
  608   FB68  48          PTSTR   PHA
  609   FB69  98                  TYA
  610   FB6A  48                  PHA
```

```
CARD #  LOC   CODE       CARD    10        20        30        40        50        60        70
  611   FB6B  A0 C6              LDY #NULL
  612   FB6D  20 73 FB           JSR PUTSTR
  613   FB70  68                 PLA
  614   FB71  A8                 TAY
  615   FB72  68                 PLA
  616   FB73  AD 00 14  PUTSTR   LDA DUARTS
  617   FB76  29 02              AND #%00000010
  618   FB78  F0 F9              BEQ PUTSTR
  619   FB7A  B9 89 FB           LDA ASCII,Y
  620   FB7D  C9 00              CMP #00          ARE WE AT THE END OF THE STRING?
  621   FB7F  F0 07              BEQ P0           YES, SO RETURN
  622   FB81  8D 01 14           STA DUARTD
  623   FB84  C8                 INY
  624   FB85  4C 73 FB           JMP PUTSTR
  625   FB88  60         P0      RTS
```

```
CARD #  LOC   CODE        CARD  10        20        30        40        50        60        70
   627  FB89              ;
   628  FB89              ; CONSTANTS AND TABLES FOLLOW
   629  FB89              ;
   630  FB89              ; THE STRING TABLE FOLLOWS BELOW.  EACH STRING HAS A SYMBOLIC OFFS
   631  FB89              ;
   632  FB89              PKCNT   = 0             PACKET COUNT MESSAGE
   633  FB89  50 41       ASCII   .BYTE 'PACKET COUNT? (00 - FF, HEX)'
   633  FB8B  43 48
   633  FB8D  45 54
   633  FB8F  20 43
   633  FB91  4F 55
   633  FB93  4E 54
   633  FB95  3F 20
   633  FB97  28 30
   633  FB99  30 20
   633  FB9B  20 20
   633  FB9D  46 46
   633  FB9F  2C 20
   633  FBA1  48 45
   633  FBA3  58 29
   634  FBA5              CRLF    = *-ASCII       CARRIAGE RETURN, LINE FEED
   635  FBA5  0D                  .BYTE  $0D
   636  FBA6              LF      = *-ASCII       LINE FEED ONLY
   637  FBA6  0A                  .BYTE  $0A,00
   637  FBA7  00
   638  FBA8              ;
   639  FBA8              PKSKP   = *-ASCII       PACKETS SKIPPED MESSAGE
   640  FBA8  50 41               .BYTE 'PACKETS SKIPPED? (00 - FF, HEX)'
   640  FBAA  43 48
   640  FBAC  45 54
   640  FBAE  53 20
   640  FBB0  53 48
   640  FBB2  49 50
   640  FBB4  50 45
   640  FBB6  44 3F
   640  FBB8  20 28
   640  FBBA  30 30
   640  FBBC  20 2D
   640  FBBE  20 46
   640  FBC0  46 2C
   640  FBC2  20 48
   640  FBC4  45 58 29            .BYTE 'PACKETS SKIPPED? (00 - FF, HEX)'
   641  FBC7  0D                  .BYTE  $0D,$0A,00
   641  FBC8  0A
   641  FBC9  00
   642  FBCA              MNADD   = *-ASCII       MONITOR ADDRESS MESSAGE
   643  FBCA  4D 4F               .BYTE 'MONITOR ADDRESS? (HEX)'
   643  FBCC  4E 49
   643  FBCE  54 4F
   643  FBD0  52 20
   643  FBD2  41 44
   643  FBD4  44 52
   643  FBD6  45 53
   643  FBD8  53 3F
   643  FBDA  20 28
```

```
CARD #  LOC   CODE        CARD  10        20        30        40        50        60        70
  643   FBDC  48 45
  643   FBDE  58 29
  644   FBE0  0D                .BYTE  $0D,$0A,00
  644   FBE1  0A
  644   FBE2  00
  645   FBE3              SCADD   = *-ASCII       SECOND ADDRESS MESSAGE
  646   FBE3  4F 50               .BYTE 'OPTIONAL SECOND ADDRESS? (HEX)'
  646   FBE5  54 49
  646   FBE7  4F 4E
  646   FBE9  41 4C
  646   FBEB  20 53
  646   FBED  45 43
  646   FBEF  4F 4E
  646   FBF1  44 20
  646   FBF3  41 44
  646   FBF5  44 52
  646   FBF7  45 53
  646   FBF9  53 3F
  646   FBFB  20 28
  646   FBFD  48 45
  646   FBFF  58 29
  647   FC01  0D                .BYTE  $0D,$0A,00
  647   FC02  0A
  647   FC03  00
  648   FC04              ERR     = *-ASCII       ERROR MESSAGE
  649   FC04  49 4E               .BYTE 'INPUT ERROR, TRY AGAIN.'
  649   FC06  50 55
  649   FC08  54 20
  649   FC0A  45 52
  649   FC0C  52 4F
  649   FC0E  52 2C
  649   FC10  20 54
  649   FC12  52 59
  649   FC14  20 41
  649   FC16  47 41
  649   FC18  49 4E 2E            .BYTE 'INPUT ERROR, TRY AGAIN.'
  650   FC1B  0D                  .BYTE $0D,$0A,00
  650   FC1C  0A
  650   FC1D  00
  651   FC1E              PRDINT  = *-ASCII       PROCESSING INTERRUPTED
  652   FC1E  50 52               .BYTE 'PROCESSING INTERRUPTED.'
  652   FC20  4F 43
  652   FC22  45 53
  652   FC24  53 49
  652   FC26  4E 47
  652   FC28  20 49
  652   FC2A  4E 54
  652   FC2C  45 52
  652   FC2E  52 55
  652   FC30  50 54
  652   FC32  45 44 2E            .BYTE 'PROCESSING INTERRUPTED.'
  653   FC35  0D                  .BYTE $0D,$0A,00
  653   FC36  0A
  653   FC37  00
  654   FC38              FINISH  = *-ASCII       PROCESSING FINISHED
```

```
CARD #  LOC   CODE        CARD  10        20        30        40        50        60        70
  655   FC38  50 52              .BYTE 'PROCESSING FINISHED.'
  655   FC3A  4F 43
  655   FC3C  45 53
  655   FC3E  53 49
  655   FC40  4E 47
  655   FC42  20 46
  655   FC44  49 4E
  655   FC46  49 53
  655   FC48  48 45
  655   FC4A  44 2E
  656   FC4C  0D                .BYTE $0D,$0A,00
  656   FC4D  0A
  656   FC4E  00
  657   FC4F              NULL  = *-ASCII
  658   FC4F  0E                .BYTE $0E,$0F,$0E,$0F,$0E,$0F
  658   FC50  0F
  658   FC51  0E
  658   FC52  0F
  658   FC53  0E
  658   FC54  0F
  659   FC55              BLANK = *-ASCII         BLANK SPACE
  660   FC55  20                .BYTE $20,00
  660   FC56  00
  661   FC57  41 4C 4C    ALL   .BYTE 'ALL'
  662   FC5A  30 31       TABLE .BYTE '0123456789ABCDEF'
  662   FC5C  32 33
  662   FC5E  34 35
  662   FC60  36 37
  662   FC62  38 39
  662   FC64  41 42
  662   FC66  43 44
  662   FC68  45 46
  663   FC6A              ENDERR = *-ASCII          ENDING ERROR
  664   FC6A  41 42             .BYTE 'ABNORMAL ENDING ERROR.'
  664   FC6C  4E 4F
  664   FC6E  52 4D
  664   FC70  41 4C
  664   FC72  20 45
  664   FC74  4E 44
  664   FC76  49 4E
  664   FC78  47 20
  664   FC7A  45 52
  664   FC7C  52 4F
  664   FC7E  52 2E
  665   FC80  0D                .BYTE $0D,$0A,00
  665   FC81  0A
  665   FC82  00
  666   FC83              ;
  667   FC83              ; INTERRUPT VECTORS BELOW
  668   FC83              ;
  669   FC83                    *=$FFFA
  670   FFFA  27 FB       VECTOR .WORD NMI          NON-MASKABLE INTERRUPT VECTOR
  671   FFFC  00 F8             .WORD RESET         RESET VECTOR
  672   FFFE  A9 FA             .WORD IRQ           IRQ VECTOR
  673   0000              ;
```

```
   CONSTANTS                                                    PAGE  20

CARD # LOC    CODE       CARD  10        20        30        40        50        60        70
  674  0000                    .END


END OF MOS/TECHNOLOGY 650X ASSEMBLY VERSION 5.1
NUMBER OF ERRORS =   0,  NUMBER OF WARNINGS =   0
```

SYMBOL TABLE

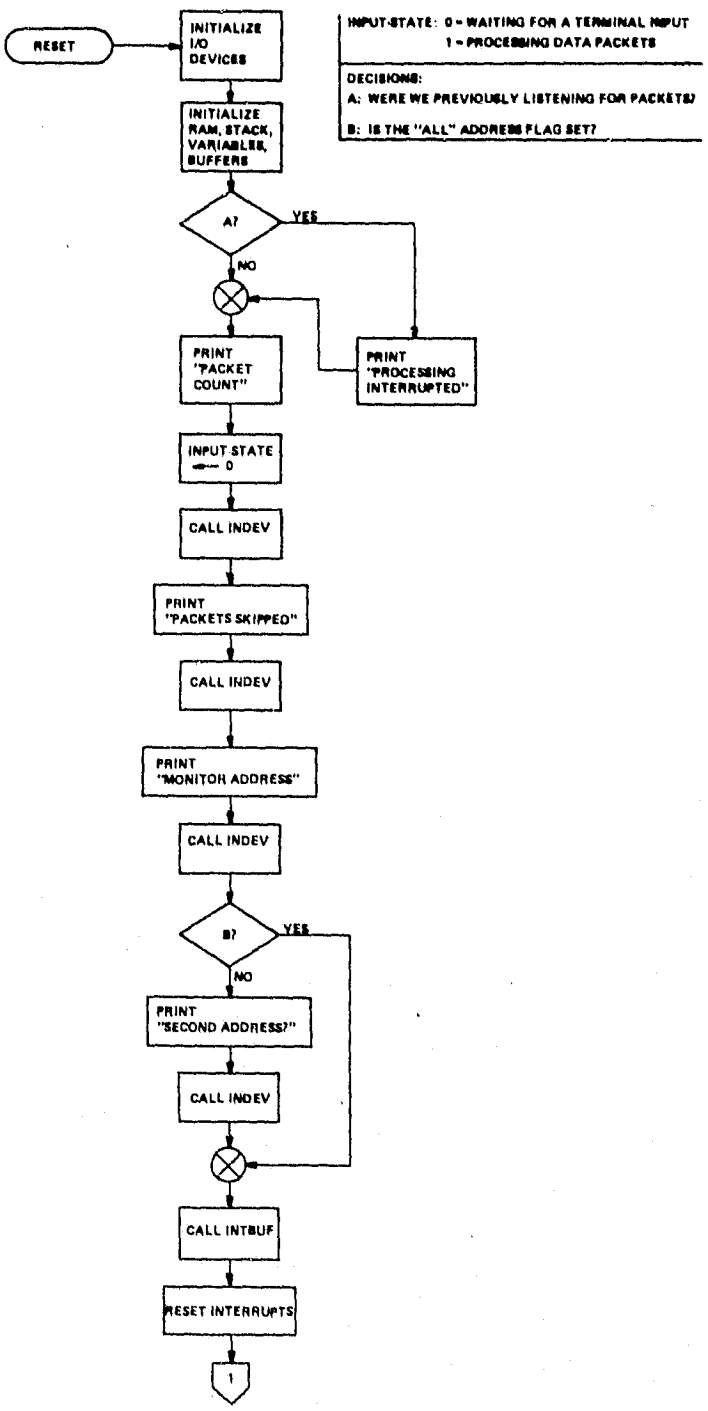
SYMBOL	VALUE	LINE DEFINED		CROSS-REFERENCES								
A1	F992	305	300									
A2	F994	306	304									
AK1	F941	259	250	256								
ALFLG	0010	39	147	291	481							
ALL	FC57	661	241									
ALLOC	F934	561	451									
ALLOC1	F936	562	572									
ALLOC2	F936	566	569									
ALPHA	F943	297	314	319								
ASCII	F949	633	619	634	636	639	642	645	648	651	654	657
			659	663								
ASKAGN	F920	249	242	290	334							
BLANK	000C	659	414									
BJFCNT	0016	60	54	63	64	66	91	104	112	115	352	571
BJFLEV	0040	19	20	123								
BJFSTK	0061	65	106	566	598							
BJFMEM	0000	19	20									
BJFFRI	F8A6	119	127									
C0	F99F	313	336									
C1	F944	325	310	317								
C2	F946	329	312									
C3	F9C5	335	331	333								
CONECT	001C	52	95	131	133	172	185	292	440			
CRLF	601C	630	129	222	422	428	430					
CRTN	F9B5	326	****									
CTON	F997	309	264									
CURVET	0008	51	458	530								
DD	F959	593	427									
DJARTD	1401	12	175	183	404	413	622					
DJARTS	1400	11	81	83	169	181	390	405	616			
ECHO	0017	45	178									
ENDERR	00E1	663	437									
ENQ	F87A	560	531									
ENQ1	F853	565	583									
ERR	0078	648	259									
FINISH	00AF	659	439									
FLEET	0008	54	278	524								
G0	F905	226	233									
G1	F907	227	237									
G2	F91B	240	247									
GCRLF	F8FE	222	213									
GETRPY	F8D4	200	186									
GLF1	F8FA	220	218									
GLF	F8F4	217	211									
GO0	F8A8	175	171									
GO1	F8C7	190	188									
GPUT	F900	223	221									
HIPTR	004B	64	99	120	370	456						
HOME2	000F	59	272	486								
HOME1	000E	57	251	285	484							
I1	F968	138	134									
I2	F88C	152	148									
IN0	F884	141	182									
IN1	F8E2	207	203									

SYMBOL	VALUE	LINE DEFINED		CROSS-REFERENCES							
IN2	F8E4	208	206								
INBC	0003	27	208	214	217	219	230	294	309	335	
INCNTR	0011	40	252	263	266	288	293				
INDEV	F89C	169	140	143	146	151	160	173	215		
INTRES	FAA7	461	452								
INTPTR	0001	26	455	457	499	510	513				
INTBC	0010	25	474	518	519						
INTSET	000A	33	447	459	492	532					
INTBUF	FA8E	447	152	159	533						
INTRET	FAA6	460	448								
IRET	F8A7	174	169								
IRQ	FAA9	467	672								
LF	0010	636	220								
LINCNT	000C	35	384	416	417	421					
LOPTR	0035	63	101	118	121	368	454				
M0	F961	275	268								
M0R	F968	277	276								
M1	F96C	282	269								
M2	F970	285	271								
MATCHA	F974	289	244								
MATCH	F919	263	231	254	258						
MLOOP	F890	158	161								
MNADD	0041	642	144								
MRET	F980	294	280	283	286						
MSGCNT	0012	41	142	279	343	434	526	528			
N10	F8DD	524	511								
N13	FAC5	484	483	485							
N16	FAF6	512	505								
N3	FACE	492	489								
N5	FAD2	494	487	491	507						
N6	FADA	498	493								
N7	FADE	500	516	520							
N8	FAOF	501	503								
N9	F804	519	517								
NEXT	001D	54	107	113	366	426	580	586	593	594	595
NINT	FAAC	474	****								
NIRET	F821	535	496	527							
NMATCH	F915	235	229								
NMI	F827	548	670								
NOECHO	F88C	184	179								
NORMAL	FA84	439	435	436							
NJARIO	0C01	10	475	512							
NJARIS	0C00	9	77	79	474	495	501	523	550	552	
NJLL	00C6	657	381	611							
OT01	F9DE	355	353								
OT1	F9CE	346	344								
OT22	F9F9	368	360								
OT2	F9E0	355	348	432							
OT21	F9F1	364	362								
OJTFRE	FA68	426	389								
OJTSET	0009	32	356	372	388						
OJTDEV	F9C9	343	158								
OJTPTR	0004	28	369	371	376	394	409				
OJTPL	0007	30	377	386							
OUTBC	0006	29	374	385	395	408					
OJTRET	FA7A	434	363								
P0	F848	625	621								

SYMBOL	VALUE	LINE DEFINED		CROSS-REFERENCES								
P<CNT	0000	652	138									
P<SKP	001F	639	141									
PRDINT	0095	651	135	193								
PTSTR	FB58	608	136	139	142	145	150	194	260	441		
PUTCH1	FA0D	381	357	424								
PUTSTR	FB73	615	130	223	382	415	423	429	431	612	618	624
PUTCH	FA16	385	419									
QEND	0034	59	95	349	355	359	367	581				
QIT	FB17	530	525									
QSTART	0053	58	94	350	358	426	587					
R0	FB1C	87	89									
RAMSIZ	0C00	17	19									
READY	F97C	292	273									
RESET	FB00	72	261	442	671							
RET	FB23	511	****									
SCADD	005A	645	149									
SKIP1	FB55	125	134									
SKPCNT	0013	42	191	282	488	490						
SNDMES	FA88	441	438									
STACKI	FB32	105	110									
STKPTR	0000	55	92	561	565	596	599					
STRING	0019	48	209	227	240	313	318	329				
T0	FA3B	405	407									
TABLE	FC5A	662	228	403	412							
TEMP0	0015	44	297	299	301	305						
TEMP	0014	43	200	202	204	207						
TEMP1	0016	45	315	324								
TRYDS	FA21	390	387	392								
VECTOR	FFFA	670	****									
WRAP	FB40	565	563	570								
WRAP1	FB46	571	564									
WRPFLG	0018	47	347	361	365	568	582					

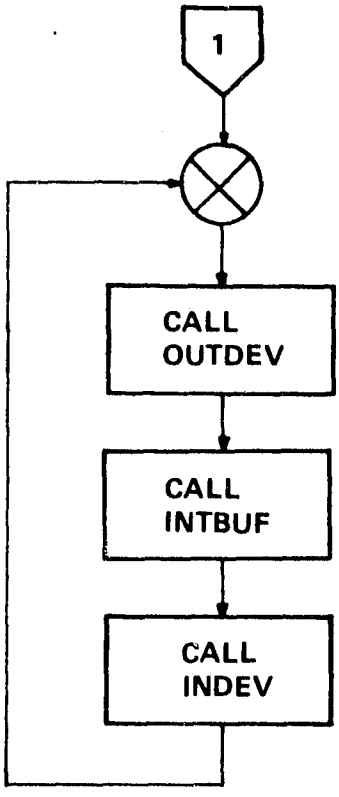
APPENDIX II

BUS LISTENER FLOW CHART


RESET
INITIALIZE I/O DEVICES
INITIALIZE RAM, STACK, VARIABLES, BUFFERS
A?
YES
NO
PRINT "PACKET COUNT"
PRINT "PROCESSING INTERRUPTED"
INPUT STATE ← 0
CALL INDEV
PRINT "PACKETS SKIPPED"
CALL INDEV
PRINT "MONITOR ADDRESS"
CALL INDEV
B?
YES
NO
PRINT "SECOND ADDRESS?"
CALL INDEV
CALL INTBUF
RESET INTERRUPTS
1
INPUT-STATE: 0 = WAITING FOR A TERMINAL INPUT
1 = PROCESSING DATA PACKETS
DECISIONS:
A: WERE WE PREVIOUSLY LISTENING FOR PACKETS?
B: IS THE "ALL" ADDRESS FLAG SET?

RESET: (CONCLUDED)

MAIN LOOP:



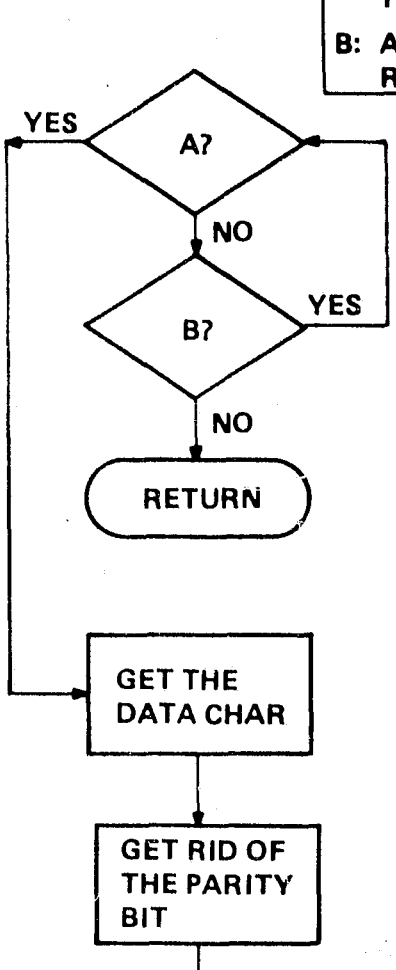
IA-50,166

INDEV:

DECISIONS:

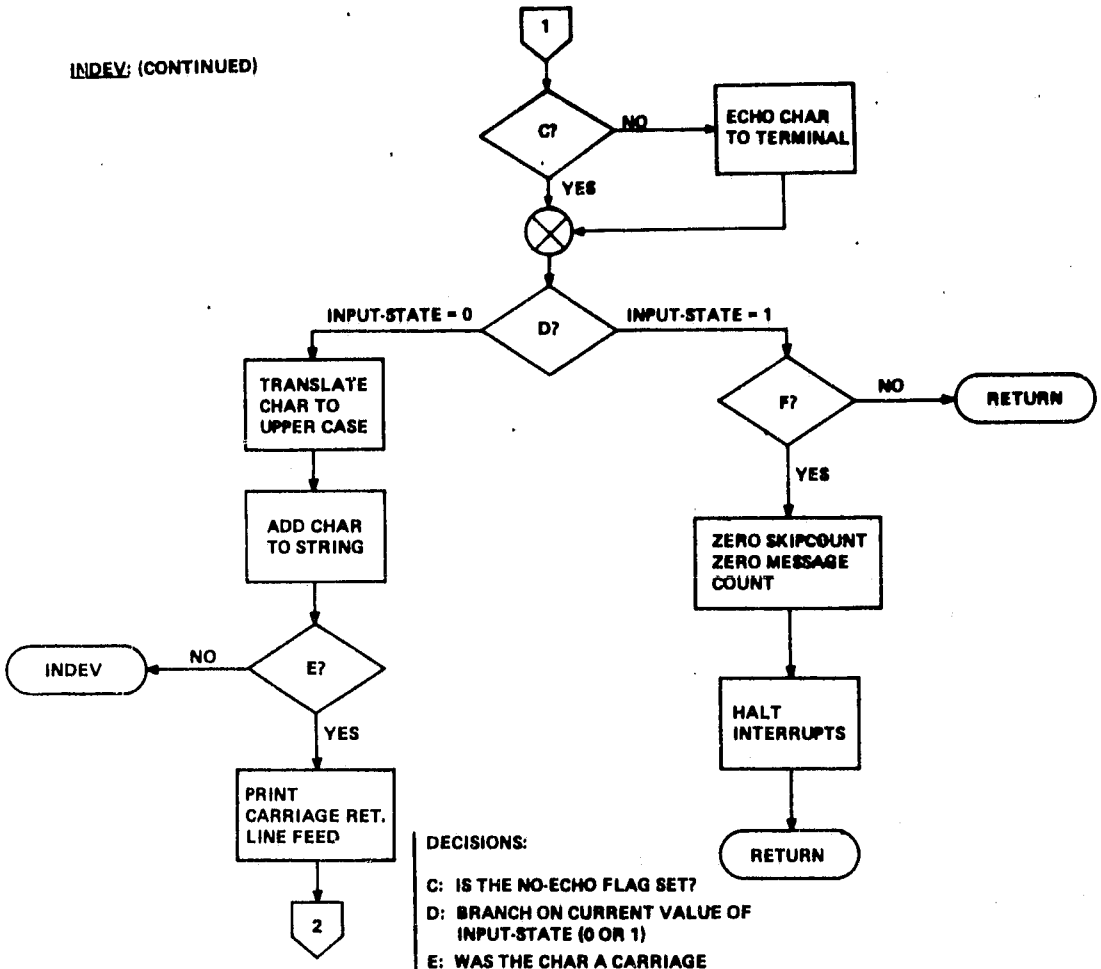
A: IS A CHARACTER READY FROM THE DEVICE?

B: ARE WE WAITING FOR A RESPONSE FROM THE USER?



IA-50,167

INDEV: (CONTINUED)



DECISIONS:

C: IS THE NO-ECHO FLAG SET?

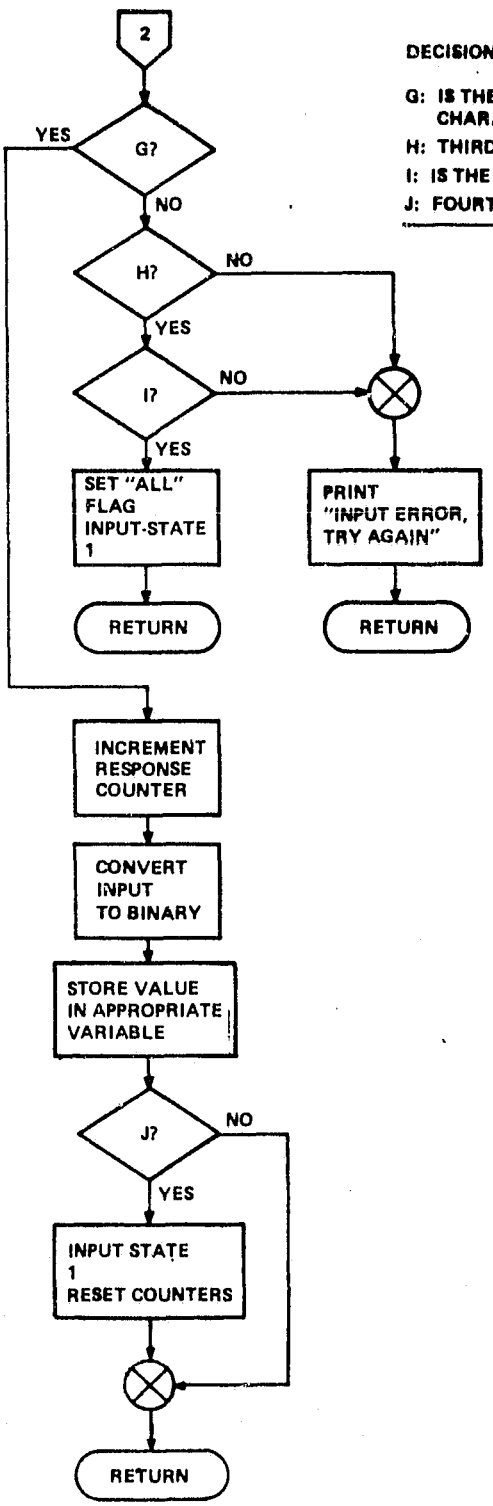
D: BRANCH ON CURRENT VALUE OF INPUT-STATE (0 OR 1)

E: WAS THE CHAR A CARRIAGE RETURN OR HAVE 5 CHARS BEEN TYPED?

F: WAS THE "ESC" CHARACTER TYPED?

IB-50,168

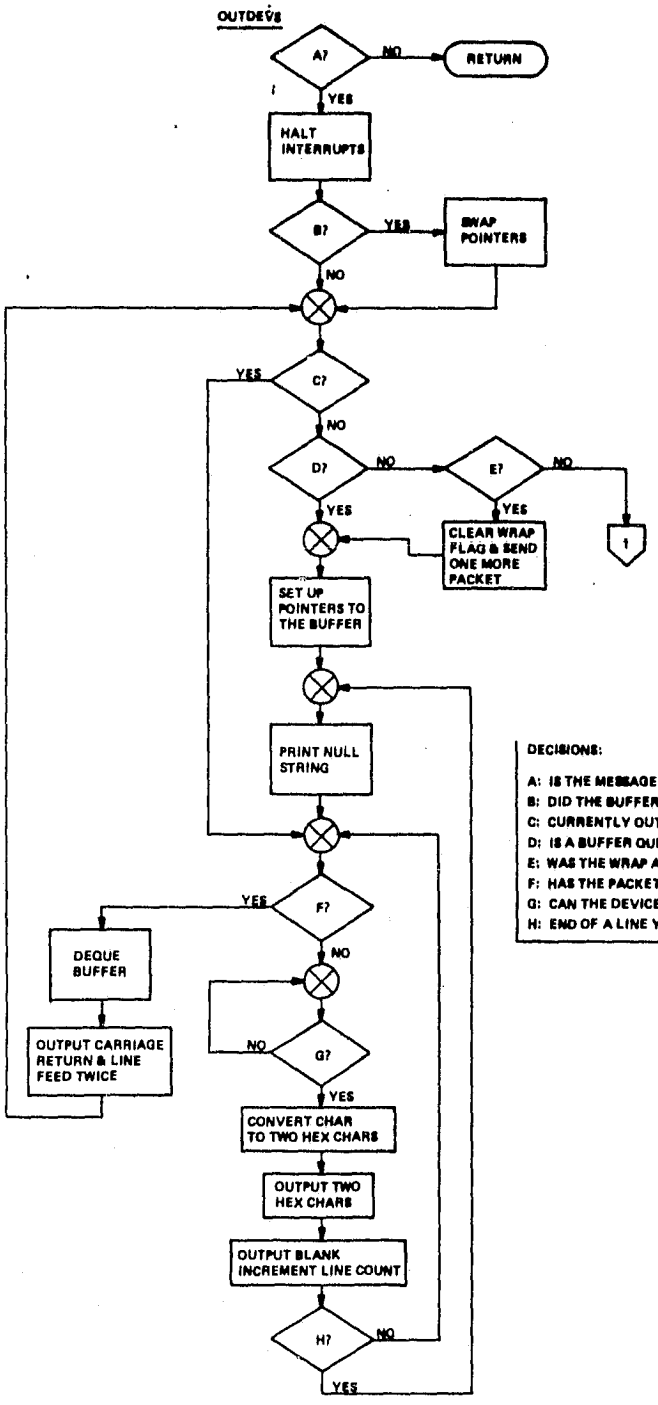
INDEV: (CONCLUDED)



DECISIONS:

G: IS THE STRING ALL HEX CHARACTERS?
H: THIRD OR FOURTH PASS?
I: IS THE STRING THE WORD "ALL"?
J: FOURTH PASS?

IB-50,169

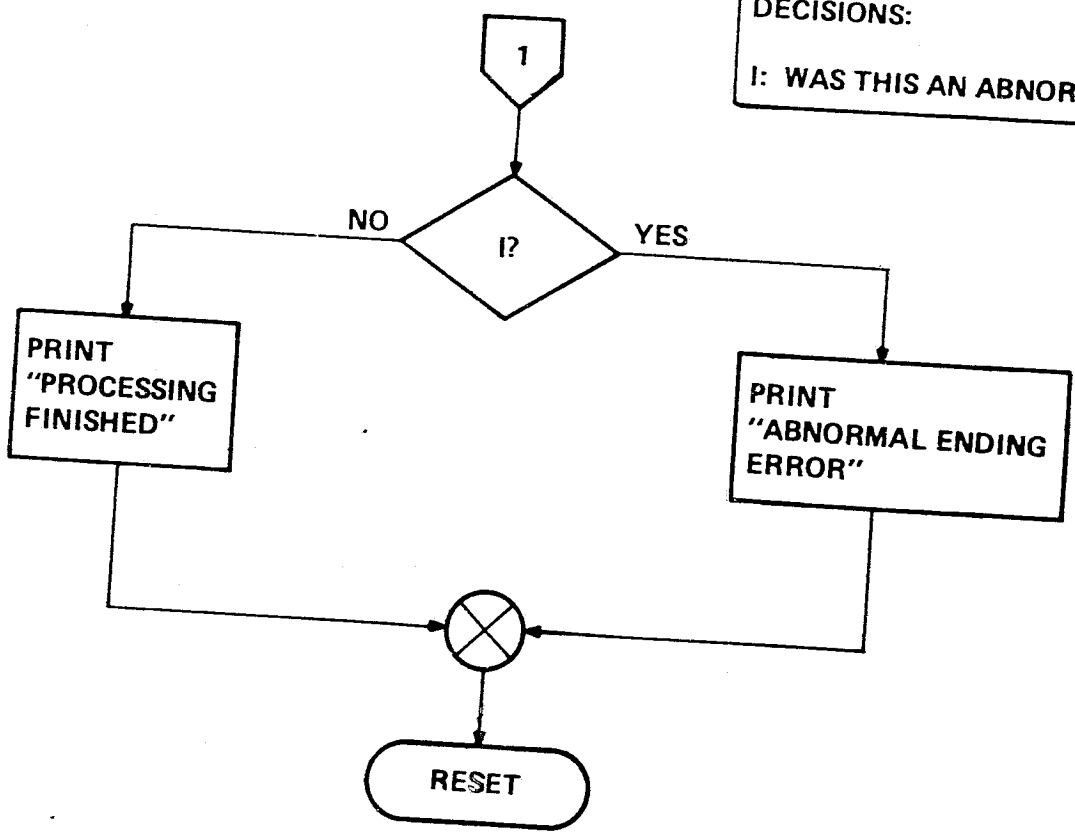


DECISIONS:

A: IS THE MESSAGE COUNT FINISHED?
B: DID THE BUFFER WRAP AROUND?
C: CURRENTLY OUTPUTTING A BUFFER?
D: IS A BUFFER QUEUED FOR OUTPUT?
E: WAS THE WRAP AROUND FLAG SET?
F: HAS THE PACKET BEEN COMPLETELY PRINTED?
G: CAN THE DEVICE TAKE A CHARACTER?
H: END OF A LINE YET?

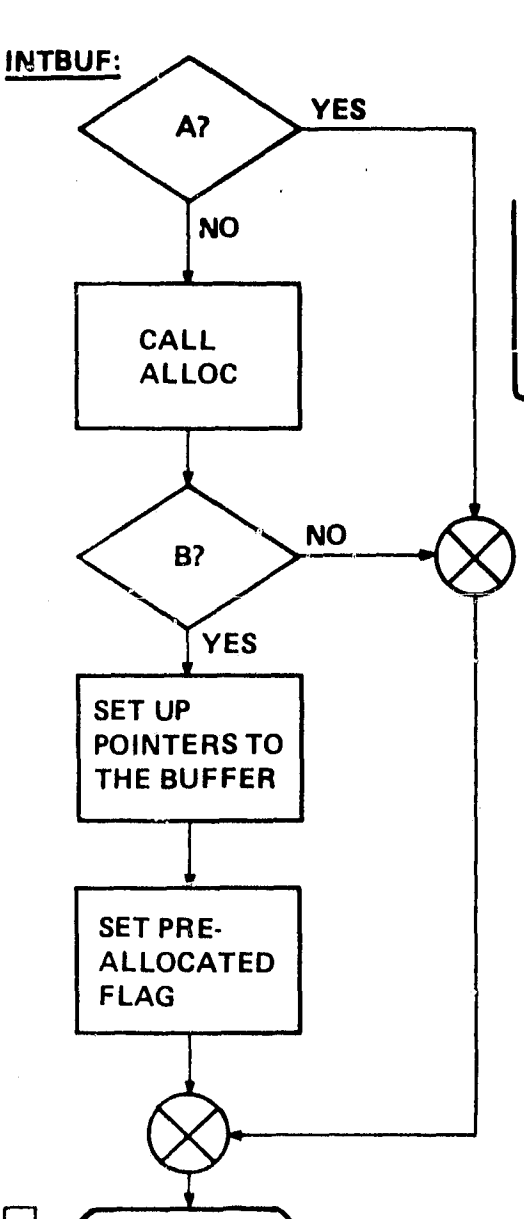
IC-50,170

OUTDEV: (CONCLUDED)



DECISIONS:

I: WAS THIS AN ABNORMAL END?

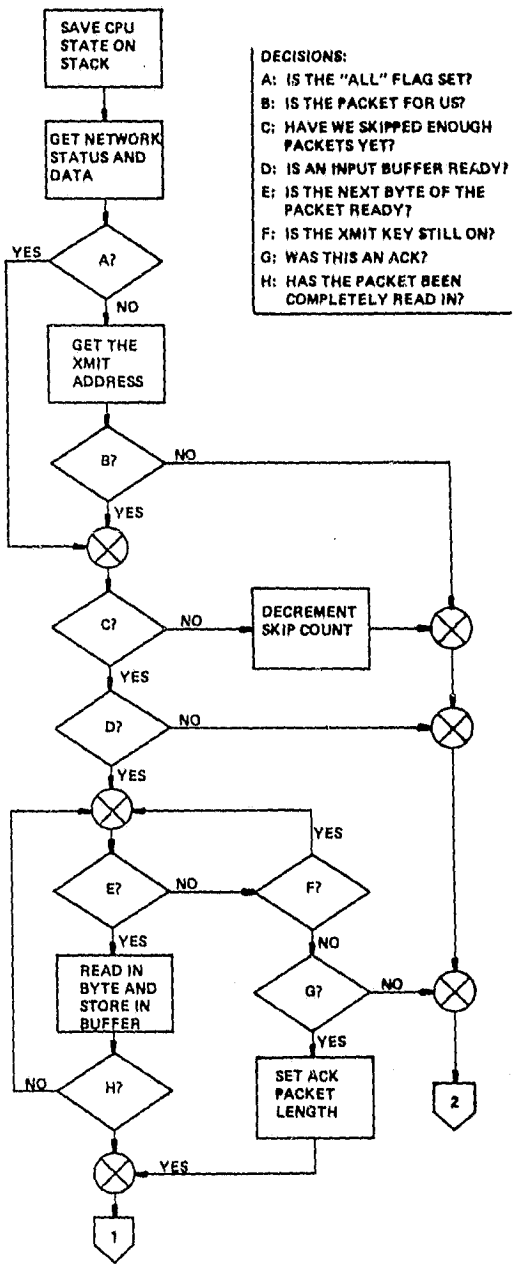


DECISIONS:

A: HAS A NETWORK INPUT BUFFER BEEN PRE-ALLOCATED?

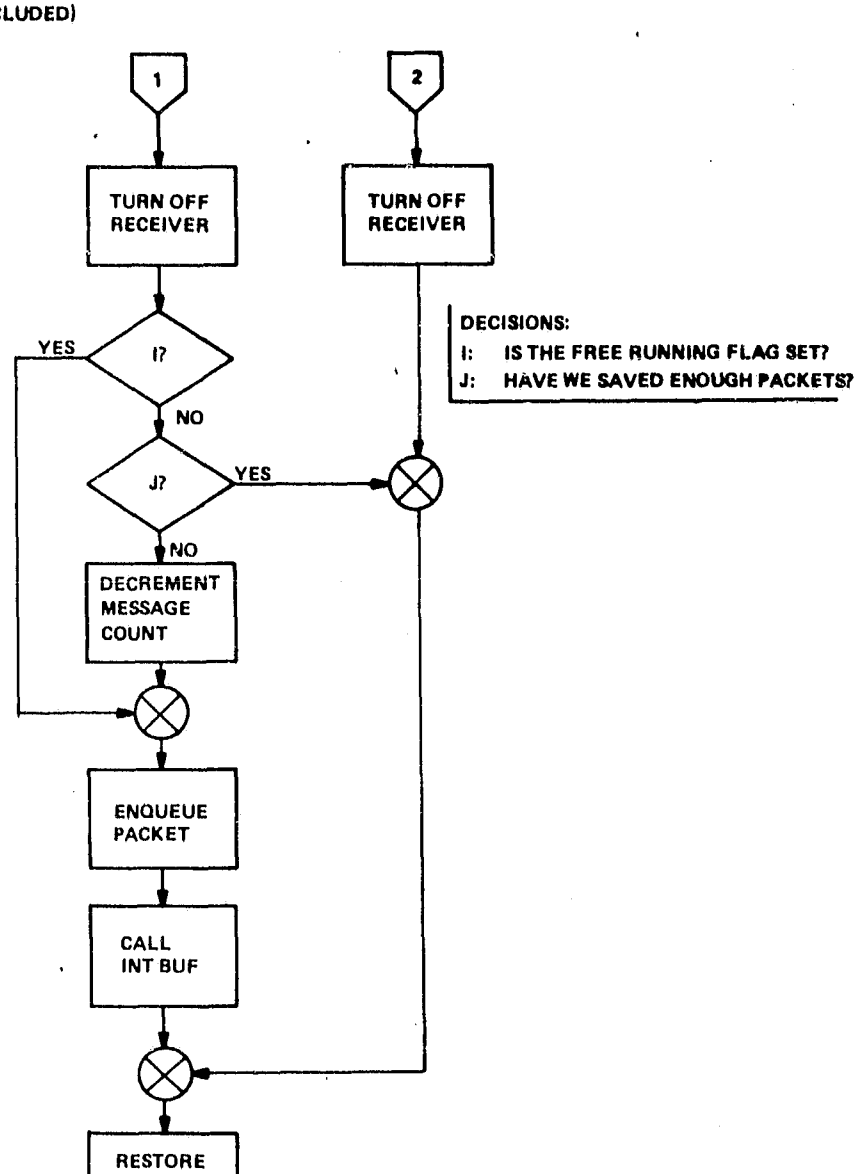
B: WAS THE ALLOCATION SUCCESSFUL?

IRQ:


SAVE CPU STATE ON STACK
GET NETWORK STATUS AND DATA
A?
YES
NO
GET THE XMIT ADDRESS
B?
NO
YES
C?
NO
DECREMENT SKIP COUNT
YES
D?
NO
YES
E?
NO
F?
YES
NO
READ IN BYTE AND STORE IN BUFFER
G?
NO
YES
H?
NO
SET ACK PACKET LENGTH
YES
2
1
DECISIONS:
A: IS THE "ALL" FLAG SET?
B: IS THE PACKET FOR US?
C: HAVE WE SKIPPED ENOUGH PACKETS YET?
D: IS AN INPUT BUFFER READY?
E: IS THE NEXT BYTE OF THE PACKET READY?
F: IS THE XMIT KEY STILL ON?
G: WAS THIS AN ACK?
H: HAS THE PACKET BEEN COMPLETELY READ IN?

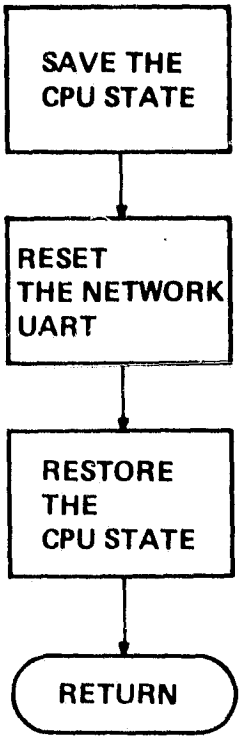
IB-50,173

IRQ: (CONCLUDED)



IB-50,174

NMI:


SAVE THE CPU STATE
RESET THE NETWORK UART
RESTORE THE CPU STATE
RETURN

IA-50,175

REFERENCES

(1) Brown, J. S. and Hopkins, G. T., Trend Monitoring System (TMS) Communications Hardware - Volume II - Bus Interface Units, The MITRE Corporation, MTR-4721 (JSC #14723), March 1979.

(2) Gregor, Paul J., Trend Monitoring System (TMS) Communications Software - Volume II - Bus Interface Unit, The MITRE Corporation, MTR-4723 (JSC #14793), April 1979.

(3) Brown, J. S. and Lenker, M. D., Diagnostic Procedures For Trend Monitoring System (TMS) Communications, The MITRE Corporation, MTR-4724 (JSC #14794), March 1979.

(4) MCS 6500 Microcomputer Family Programming Manual, MOS Technology, Inc., Norristown, Pennsylvania, January 1976.

(5) "MC 6850 Asynchronous Communications Interface Adapter (ACIA) Data Sheet," Motorola Semiconductor Products, Inc., Phoenix, Arizona.

(6) "MCS 6522 Versatick Interface Adapter Data Sheet," MOS Technology, Inc., Morristown, Pennsylvania, November 1977.