Rochester Institute of Technology
RIT Digital Institutional Repository

Theses

5-1-1982

An Investigation and Implementation of Recommendation X.25 on a Motorola M6800 Based Network

Louis Nyerges

Follow this and additional works at: https://repository.rit.edu/theses

Recommended Citation
Nyerges, Louis, "An Investigation and Implementation of Recommendation X.25 on a Motorola M6800 Based Network" (1982). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.

ROCHESTER INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

AN INVESTIGATION AND IMPLEMENTATION OF RECOMMENDATION X.25 ON A MOTOROLA M6800 BASED NETWORK

A Thesis submitted in Partial Fulfillment of

Master of Science In Computer Science Degree Program

WRITTEN BY: LOUIS D. NYERGES

APPROVED BY:

ROY S. CZERNIKOWSKI (ADVISOR)

KENNETH A. REEK

JOHN L. ELLIS

MAY, 1982

TABLE OF CONTENTS Page

TABLE OF CONTENTS (continued) Page

ABSTRACT

A subset of the Recommendation X.25 protocol is implemented on a two node network. Each node accepts and translates data from the user in a sequential fashion into the appropriate protocol structure for transmission to another node. Each node has the capability of full duplex operation.

The hardware is built around a Motorola M6800 microprocessor employing a minimum of 4096 words of random access memory and two asynchronous ports: one for communication with the user and the other for communication with the network. The software on each node contains the protocol programs and a multitasking monitor which is used to buffer the data bytes between the protocol software and the two communication ports. The multitasking monitor is also used to time slice among the seven tasks which implement the protocol structure.

This paper illustrates in some detail the structure of Recommendation X.25. The subset of the packet types which are implemented is also examined. Finally, a scenario of the procedure for information exchange between one user and another is discussed in detail.

ACKNOWLEDGEMENTS

I wish to thank the many people who have contributed their efforts in helping me make this thesis possible. To my advisor, Dr. Roy S. Czernikowski for his support throughout the past two years and to the other members of my committee who have given their time, suggestions, and comments for the improvement of this paper, I sincerely wish to thank you. I also would like to thank the department secretary at the Computer Engineering office, Kathy Ozminkowski, for her help in transferring information between Dr. Czernikowski and me.

During the final stages of development, it was necessary to obtain a computer terminal for the numerous listings which I had made and a way of transporting the information between NTID where the terminal was located and my apartment where the development took place. The Department of Communication Support loaned me the equipment necessary to facilitate my efforts. In addition, the School of Computer Science and Technology loaned me one of their terminals so that I could complete the work on the thesis.

Finally, I wish to express my appreciation to my family and friends who have put up with me for the past two years while I did little else but work on this project.

PREFACE

Before the explanation of the implementation of Recommendation X.25 begins, it is important to present some background information. The research for this thesis began officially in the Spring of 1980, with some preliminary work being done several months earlier.

The original thesis proposal called for a more detailed system than that presented here. The proposal which was presented called for a series of three Data Termination Equipment (DTE) nodes which would communicate among themselves through the Recommendation X.25 protocol structure. These nodes were to have been connected with each other through a central node known as a switch. Communication between each of the users and their respective DTE nodes were to have used the Rochester Institute of Technology Local Protocol, (RITLP). All the software development for both protocols and the network switch were also proposed. It became apparent in the early stages of development that the original proposal was excessive in length and modifications were in order.

The revised proposal called for a two node system consisting of a single DTE communication node connected to a single Data Circuit-Termination Equipment (DCE) node. A user wishing to communicate with another user would transmit ASCII characters either serially or in block mode into its respective DTE node via an ASCII terminal. The node would then build the necessary packets for transmitting the characters. The alternate node or the DCE node would receive and decode the protocol information and issue the characters to the CRT terminal connected to the

DCE node. The system was created for full duplex operation which means that each terminal or user could transmit and receive simultaneously. This paper explains the theory and implementation of the revised proposal, including a detailed scenario of a packet transfer.

The hardware logic had already been developed to some extent. The original proposal called for the programs to be burned into Erasable Programmable Read Only Memories (EPROMs) and a working model be demonstrated. Since the hardware was untested, this too, was dropped. Instead, the software for both the DTE and the DCE communication nodes were developed and implemented on the author's personal system.

During development, problems had arisen which delayed the completion of this thesis. There were four components to the system: a CRT terminal, line printer, dual floppy disc unit, and the processor. Each of these units had failed at least once during the development and implementation of the software. The author suggests caution to anyone who may be considering a project such as this on a small development system; hardware problems do develop and an expansion factor must be introduced into the development schedule.

CHAPTER I

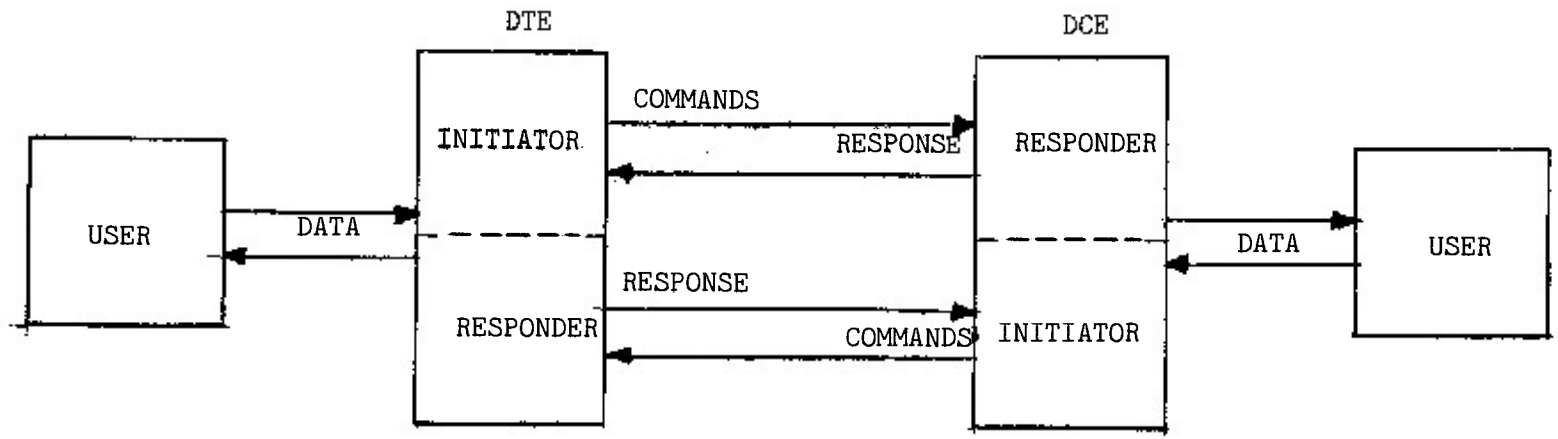
INTRODUCTION

Recommendation X.25 is a data transmission and reception protocol used in communicating between two points on a shared network. The two points are known as the Data Termination Equipment (DTE) which is located at the local site or at the host computer and the Data Circuit-Termination Equipment (DCE) which is at the remote site. The communication link between the DTE and the DCE nodes is normally a telephone access channel, either directly linked (using a four wire system) or switched (using a two wire system), communicating in a synchronous fashion in full duplex mode. Care must be taken when contracting with a telephone company for a totally full duplex line; some long distance and some local communication exchanges are not truly full duplex. Half duplex links may exist between the two nodes and communication problems will result.

Recommendation X.25 was adopted on March 2, 1976, by Study Group VII of the International Telephone and Telegraph Communication Committee, (CCITT). The paper entitled "Interface Between Data Termination Equipment and Data Circuit-Termination Equipment For Terminals Operating in the Packet Mode on Public Data Networks" was adopted throughout the United States and around the world. Under the auspices of the CCITT, Recommendation X.25 was developed jointly by the Centre Commu d'Etudes de Television et Communication of the French Post and Telecommunications

(France), Nippon Telegraph and Telephone Public Corporation (Japan), The Computer Communication Group of the Trans-Canada Telephone System (Canada), the United Kingdom Post Office, and the Telenet Corporation (USA).

The programs which appear in the appendix represent a two node structure, shown in block form in Figure 1. One node is represented by the DTE mnemonic and the other node is represented by the DCE mnemonic. For simplicity, the system can be considered to be two entirely separate communication links, one transmitting from the DTE node to the DCE node and the other transmitting in the opposite direction. This paper treats the DTE as the local node and the DCE as the remote node or site. In the DCE communication software code, the reverse is true; i.e., the DCE is the local node and the DTE is the remote node or site.



OVERALL SYSTEM BLOCK DIAGRAM

FIGURE 1

CHAPTER II

SYSTEM DESCRIPTION AND TECHNOLOGY

HIGH LEVEL SYSTEM OVERVIEW

Two ASCII terminals are used as the source and destination transducers for transmission and reception purposes in this project. Each terminal communicates with its respective node asynchronously with a transmission rate of between 300 and 1200 baud. Since the terminals operate in full duplex mode, characters generated by the keyboard will not be displayed on the terminal's screen; i.e., the keyboard and the CRT terminal screen are separate devices.

The DTE node receives the characters sent by the keyboard of its terminal and builds a packet of information by collecting the input data until the desired character quantity is reached. The size of the packet is determined by the Input routine and may vary between one and 128 bytes. To simplify viewing the data, only ASCII characters may be transmitted in this implementation. Normally, a host computer would be substituted for the terminal and allow unrestricted binary information to be transferred in packets containing up to 1024 bytes of information.

The communication channel used in this thesis is a RS-232C standard link. A voltage level of -12 volts is called a mark and represents a logical zero or off state, and +12 volts is called a space and represents a

logical one or on state. When there is no data transmission occurring, the communication link is in the off state. Each eight bit byte is framed between a start bit and a stop bit. This ten bit transmission pattern for each byte of information is sent at a data rate of 1200 baud for this project.

Recommendation X.25 calls for a synchronous transmission link in which bytes are transmitted without the start and stop bits. When the data train terminates, sync bytes are transmitted keeping the receiver synchronized with its transmitter. In both cases a modem is used to transmit the data. However, for the purposes of this thesis, the two ports are directly connected without modems.

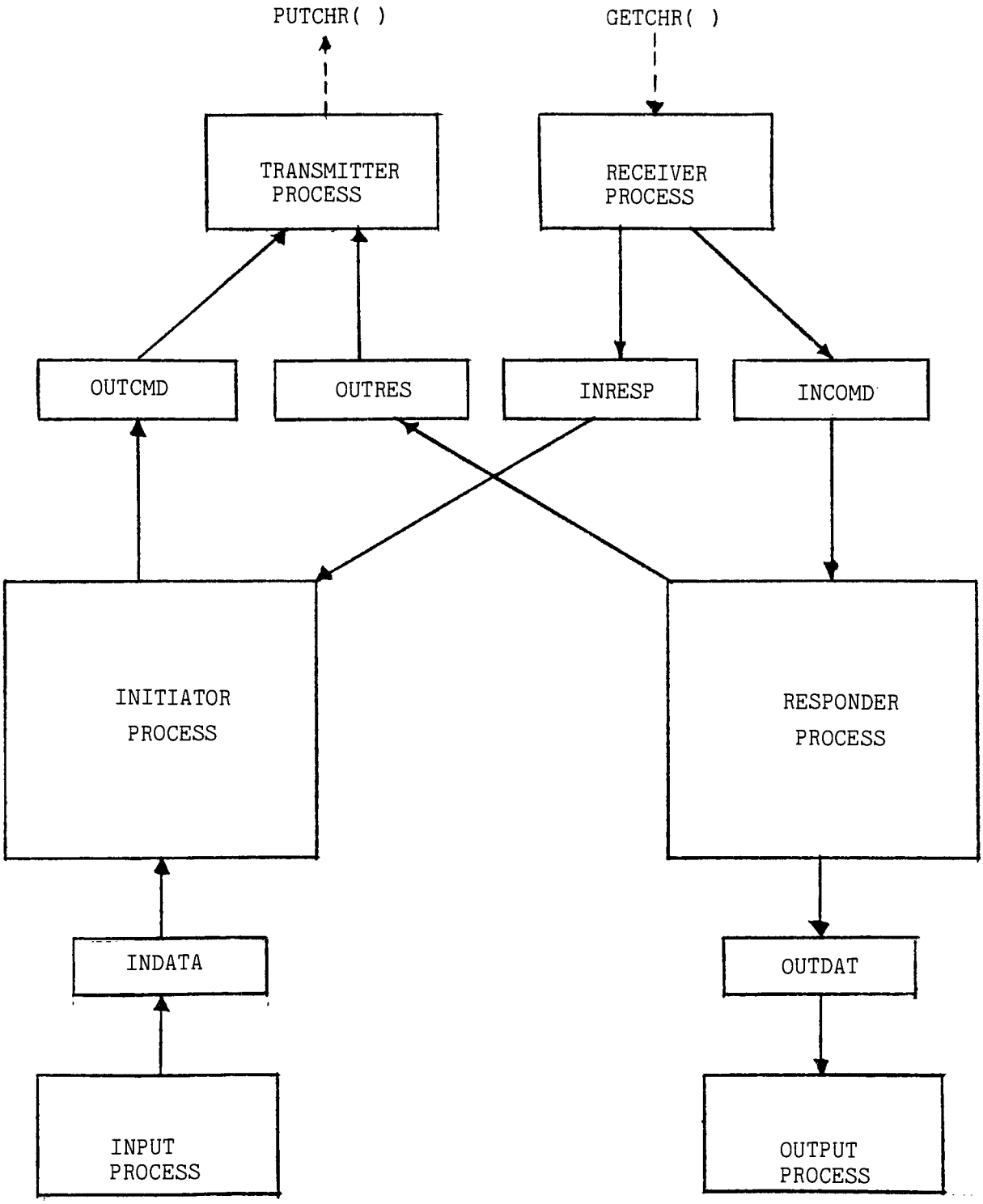
The DCE communication node receives the entire packet sent by the DTE node and stores it in one of its selected buffers. The entire packet is collected before any of its contents are processed. After the packet is processed, the information is transmitted to its user and an acknowledgement is returned to the DTE communication node. Once the CRT terminal has accepted the entire train of data bytes, the DCE buffer which held the data is freed.

APPROACH AND ORGANIZATION

IPC Buffers

The logic for each communication node is implemented as seven distinct processes. The seven are the Initiator, Responder, Transmitter, Receiver, Input, Output, and Timer processes. The tasks operate independently of each other through the aid of a multitasking monitor and communicate among themselves through Interprocess Communication (IPC) buffers. Figure 2 illustates six tasks along with the communication structure employing the IPC buffers. The Timer process communicates only with the Initiator and does not use IPC buffers. It is not represented in the diagram.

Each IPC buffer consists of two eight byte holding areas which are named the Input and Output buffers (not to be confused with the Input and Output processes). When a process wishes to transfer information to another process, the initiating process will load the Input buffer with the appropriate information and validate it. By convention, it is the responsibility of the receiving task to accept the information. The receiving process will accept the information by flipping or transposing the buffers within the buffer pair. This action presents a new, empty buffer to the initiating process and presents the received information to the destination process. The receiving task may empty and process the information at its own pace. Once the Input buffer is filled again, it will not be exchanged with the Output buffer until the destination process has empied and released it. The information presented to the destination



OVERALL COMMUNICATION NODE BLOCK DIAGRAM
FIGURE 2

IPC buffer pair will be identical to the information presented to the source IPC buffer pair.

Timer Process

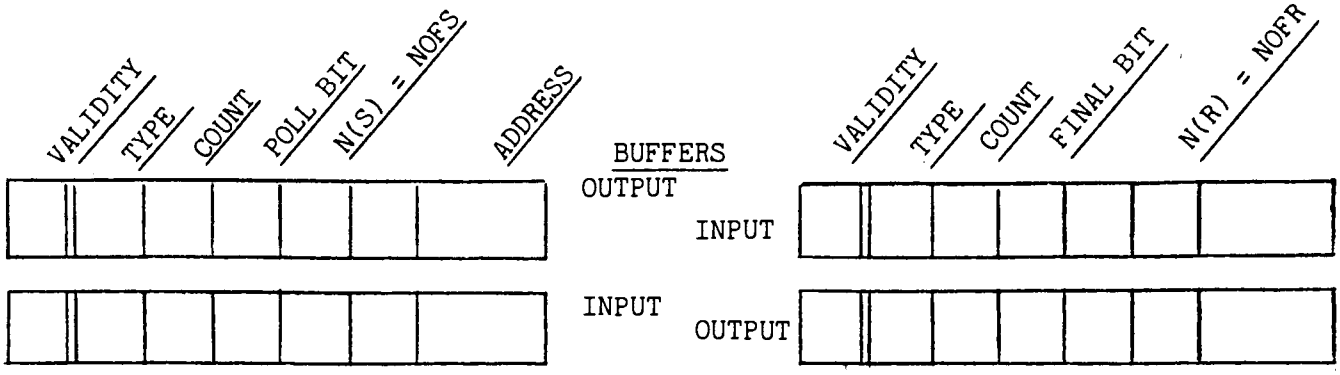
The Timer task is a process which counts clock cycles. It is used by the Initiator process to time outstanding packets (packets which have been transmitted but which have not been acknowledged). The purpose of the task is to inform the Initiator process that a given amount of time has expired so that the Initiator process can take appropriate action. The Initiator sets a time parameter and readies the Timer task, which counts the time parameter down until it reaches zero, when it sets a timeout flag. The Timer may be terminated at any time by the Initiator process thereby inhibiting the count down procedure. The Initiator process enters the Time Recovery state if a timeout occurs and the timeout flag is set.

Initiator Process

The Initiator process administers and receives packets of information and accepts incoming data from the user. It is the responsibility of the Initiator process to recover from abnormal situations and to keep the remaining processes running properly. The IPC buffers and the Input table associated with the Initiator process is illustrated in Figure 3.

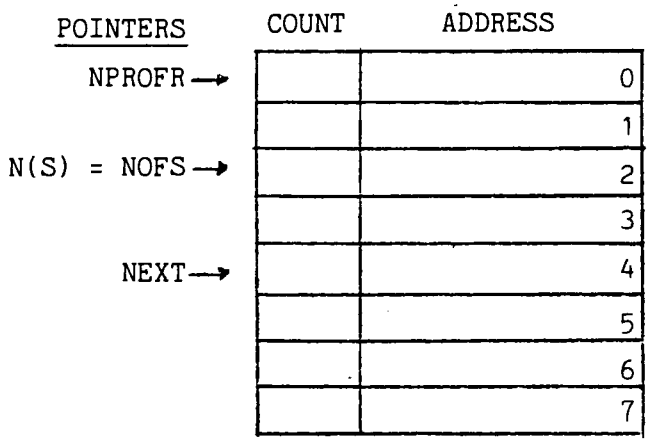
The Initiator process receives the data from the INDATA IPC buffer pair and sends it to the Transmitter process for transmission through the OUTCMD IPC buffer pair. Response packets are received from the Receiver task via the INRESP IPC buffer pair. The Input table is used by the Initiator process to queue the data before it is given to the transmitter process. The normal sequence of events in the Initiator process is to receive the incoming data and queue it, dequeue a data packet from the Input table and send it to the Transmitter process, and accept response packets which acknowledge outstanding data packets.

For simplicity (although it is not implemented as such), there are three tasks within the Initiator process. The duty of the first task would be to accept the incoming data and queue it in the Input table. However, the data would not be accepted if the Input table were full. In this case, the Input process would cycle indefinitely until an empty buffer was found. The second task would have the responsibility of dequeuing the data from the Input table, framing it with the packet information, and sending it to the Transmitter process via the OUTCMD IPC buffer pair. This task would not send the packet unless there were data in the Input table or the OUTCMD

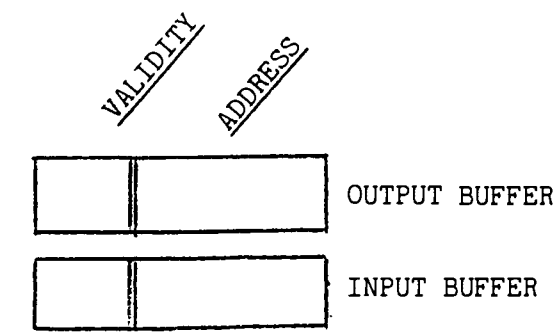


OUTCMD IPC BUFFER PAIR

INRESP IPC BUFFER PAIR



INPUT TABLE



INDATA IPC BUFFER PAIR

INTERNAL ARCHITECTURE OF THE INITIATOR PROCESS

FIGURE 3

IPC buffer was invalid or empty. The third task and the largest of the three would be responsible for the recognition of the response packets and for taking the appropriate action based on the contents of those packets. This particular task would have the power to place the Initiator process into one of several possible states, representing the conditions which have occurred because of the receipt of certain response information or the lack of information. The normal condition for the Initiator process is the Normal state. In this state, data packets are initiated and acknowledged affirmatively. The task that dequeues the data for the purposes of packet transmission only retrieves the data from the Input table. It is the responsibility of the third task to declare that the previously transmitted data in the Input table has been acknowledged affirmatively and may be deleted. Seven packets may be queued and transmitted, but if they are not acknowledged, the Initiator process cannot issue any additional packets.

After each packet is transmitted the Timer process is initialized and readied. If the Timer process sets the timeout flag, the Initiator will enter the Time Recovery state. It is assumed that a communication problem caused the timeout, so the Initiator retransmits all of the outstanding data packets in hope that at least one affirmative acknowledgement will be received. This procedure will be repeated up to four times if no affirmatively acknowledgement has been received, at which time the Initiator process assumes that the communication link is in the down state. At this point, the Initiator will try to bring the link up by transmitting a restart command. Again, four attempts will be made to bring the link up. In both cases, Recommendation X.25 requires twenty retries before giving

up. In this project, the value four was used to easily debug the software. Any number could be used.

Before the data packet has been transmitted, its information is saved in the Input table for the possibility of a retransmission situation. This condition occurs if the Initiator process assumes the link is down or if the Initiator receives a negative response packet. In the later case, a specific data packet is requested and is retransmitted by the Initiator. To do this, the Initiator process will enter the Host Rejection state and the packet in question and any outstanding data packets which follow it will be retransmitted. When an affirmative acknowledgement is received for all of the retransmitted packets, the Initiator process reverts to its Normal state.

It is possible that the receiving node cannot receive another packet of information. This condition could occur at times that the sender cannot predict. If this happens, the Initiator process will receive a different type of affirmative acknowledgement which requests that the Initiator temporarily suspends transmission. The Initiator process will then enter the Host Busy state. While in this state, response and data packets may be accepted but they will not be transmitted. A normal affirmative acknowledgment packet causes the Initiator to reenter the Normal state.

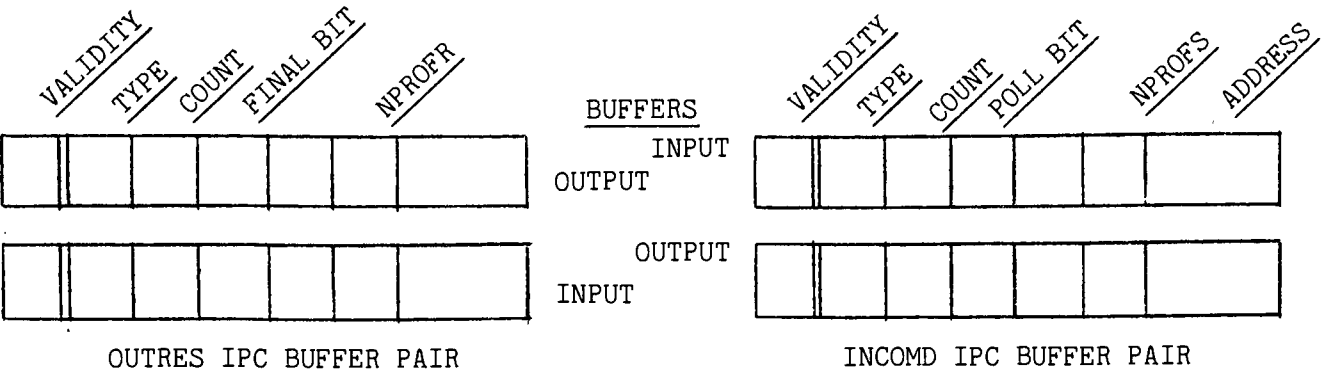
Responder Process

The complement process for the Initiator task is the Responder process, which accepts the data packets from the Initiator process and acknowledges their receipt. The Responder process must also transfer the data packets to the Output process in a logical order. The IPC buffers and the Output table associated with the Responder process are illustrated in Figure 4.

Received packets are accepted through the INCOMD IPC buffer pair. Packets that contain data are queued in the Output table. If the Responder process is in the Normal state, an affirmative acknowledgement will be generated for every received packet. The response information is passed to the Transmitter process through the OUTRES IPC buffer pair. The data is dequeued and transferred to the Output process via the OUTDAT IPC buffer. The normal procedure for the Responder is to accept and queue each incoming information packet, transmit an acknowledgement for that specific packet, and dequeue a data packet and give it to the Output process.

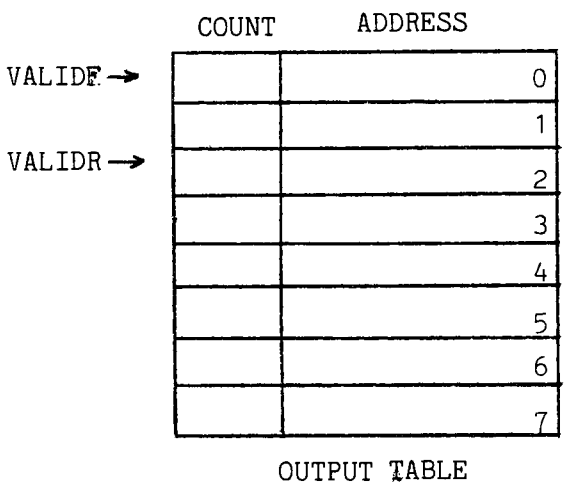
For simplicity, the Responder process may be considered as two separate tasks even though it is not implemented in this way. The first task has the responsibility of dequeueing the data and transferring it to the OUTDAT IPC buffer. Data will not be transferred unless the Output table is partially filled or the Input buffer of the OUTDAT IPC buffer pair is invalid or empty. Unlike the Initiator process, when this task dequeues data it also deletes it from the Output table.

The second and larger of the two tasks is responsible for placing the

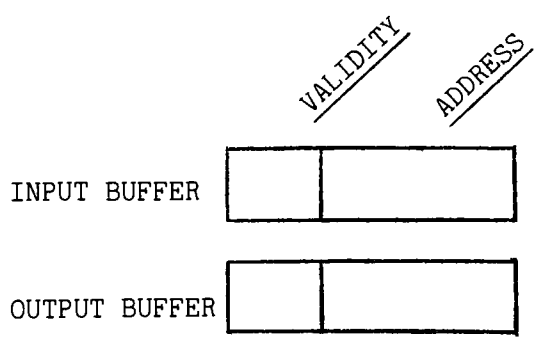


OUTRES IPC BUFFER PAIR

INCOMD IPC BUFFER PAIR



OUTPUT TABLE



OUTDAT IPC BUFFER PAIR

INTERNAL ARCHITECTURE OF THE RESPONDER PROCESS

FIGURE 4

data into the Output table in a logical order. This task must also acknowledge the received data packets and place the Responder process in the proper state. While in the Normal state, each received data packet is queued by the second task, which transmits an affirmative acknowledgement.

The Responder process is considered a passive task. This is due to the fact that it is not under any time requirements, unlike the Initiator which takes action when a timeout occurs. The Responder process will transmit a response packet only after receiving a packet. However, this does not imply that every received packet will be acknowledged.

Assume that the Responder process has received an erroneous data packet. It will transmit a negative acknowledgement and delete the packet. Further, assume that the sending Initiator process has several outstanding packets and the packet which was rejected was the first one transmitted. This implies that the Responder process received the first packet, declared it in error, and generated a negative acknowledgement. Before the Initiator process can retransmit the erroneous data packet, the Responder task will have already received the rest of the data packets, and must reject them. Since one negative acknowledgement has already been transmitted and the Responder process knows that the Initiator will retransmit not only the rejected packet but also all remaining outstanding packets in sequence, the Responder merely deletes the remaining packets without sending any acknowledgements. The Responder process enters the Node Rejection state in which all of the original packets which followed the rejected one are disposed of. The Normal state will not be reentered until another copy of the rejected data packet has been received and deemed

valid.

The Responder can also acknowledge a packet with the stipulation that the Initiator process should temporarily cease transmission. This is known as the Node Busy state, and occurs when the Output table is filled to capacity. If additional packets are transmitted by the Initiator process, they will be discarded without acknowledgement.

Transmitter Process

The transmitter process aids the Initiator and Responder tasks by providing them with the ability to easily transmit packet information. The Transmitter process also frames the data given to it into legal packets so that the Initiator and Responder processes do not have to know about the internal structure of the packets. Only as much data as will fit into a packet can be given to the Transmitter process at any one time, however.

When the Initiator process wishes to transmit a packet, it transfers the data to the INCOMD IPC buffer. The Responder task may wish to transmit a response packet independently from the Initiator process. It does this by transferring the appropriate packet information to the OUTRES IPC buffer. Each Input buffer of the Transmitter process is selectively scanned and any valid packet is transmitted to the network.

Even though the Transmitter process is not implemented in this fashion, it could be considered as two separate tasks, each transmitting packets of information. One task would be responsible for the receipt and transmission of a command packet which originates from the INCOMD IPC buffer pair and the other task would be concerned with the receipt and transmission of a response packet which originates from the INRESP IPC buffer pair. In the actual implementation of the Transmitter process, each task performs its function entirely before releasing control to the other task.

Receiver Process

The complement of the Transmitter process is the Receiver task. This task has the responsibilities of accepting packets which have been sent to it from its corresponding Transmitter process and decoding the framed packet information. The Receiver performs these tasks so that the processes that use the data need not be concerned with the structure of the packets. There are two basic types of packets which this process can receive, Command and Response packets, which are sent to different processes.

Command packets, which originate in the Initiator process, are decoded and sent to the Responder process through the INCOMD IPC buffer pair. Conversely, Response packets from the Responder process are decoded and sent to the Initiator process through the INRESP IPC buffer pair.

The Receiver process accepts an entire framed packet before beginning to process it. Packets whose length exceeds 140 bytes are ignored, and the partially collected packet is discarded. This is due to the fact that the buffers can hold no more than 140 bytes. This task also has the capability of aborting a reception which is already in progress. The partially collected packet is discarded and a new frame is accepted. This procedure may occur at any time during the course of receiving a packet. This capability is a function of Recommendation X.25 and is used to terminate a lengthy packet due either to a transmission fault or by an extra DLE character sequence sent by the Transmitter process. Neither situation can occur in this implementation.

The Receiver process also checks packets for validity. Before it passes the information from a packet to its destination, the Receiver process calculates a checksum and compares it to the received checksum value. If an error exists, a flag will be set which will accompany the data to the destination process. The Receiver process itself does not act upon any errors; this is left to the destination task.

The Transmitter and Receiver processes and the Initiator and Responder processes constitute a logical data path for transmission and reception purposes. The remaining two processes, Input and Output, accept data from and present data to the user, respectively.

Input & Output Processes

The Input and Output processes are user dependent; they are the interface between the communication node and the user. A common format is used to transfer data between the Input and the Initiator processes and between the Responder and the Output processes. It is the responsibility of the Input and Output processes to convert the data between this format and the user's format.

Originally, the 'user' was to have been the RITLP protocol. The Input process would have accepted incoming RITLP packets and passed the data to the Initiator process through the INDATA IPC buffer pair. The Output process would have converted the data received through the OUTDATA IPC buffer pair to the format suitable for the RITLP protocol. Because this protocol was not implemented in time for this project, a different technique for data interchange was developed.

A keyboard terminal is used as the input device for each communication node. With this configuration, the Input process accepts the incoming data as it is sent from the keyboard and builds a data cluster, which is nothing more than a group of data bytes of a specified quantity. As mentioned earlier, a packet may have a data length of 1 to 128 bytes. We shall define this data field of the packet to be a data cluster.

In order to provide a continuous stream of input data, the Input process obtains bytes from a character generator routine rather that from the keyboard. Using the Input process with the character generating routine excercises the protocol under worst case conditions. Only

printable ASCII characters are generated, and they are sent to the Initiator process at the fastest possible rate.

The data packet is accepted, as a whole, from the Responder process by the Output task through the OUTDAT IPC buffer pair. Once a packet has been accepted, the individual bytes are transmitted to the user. The Output process will not accept additional packets until all of the data bytes have been accepted by the user.

PACKET DESCRIPTION

A packet is the basic unit that is transmitted between two communication nodes. Each packet consists of a header that determines the packet type and a data field. Recommendation X.25 supports a wide variety of packet types of which eight are implemented in this project. These packets are classified in Figure 5.

Recommendation X.25 supports two general classifications of packets, Command and Response packets. There are two types of Command packets, Information Transfer packets and Unnumbered packets. The Information Transfer packet, also known as the data packet, carries data byte information between the sender and receiver.

There are two types of Response packets which are Supervisiory and Unnumbered packets. There are three types of Supervisory packets. RR, RNR, and REJ, are the only packets which are used to acknowledge receipt of data packets. Each of these packets has its own unique meaning. The RR or Receiver Ready response packet affirmatively acknowledges receipt of data packets. It is sent by the Responder process when a valid data packet has been received. It also informs the sender that the receiver can accept another data packet.

The RNR or Receiver Not Ready packet also affirmatively acknowledges receipt of a data packet, but indicates that additional packets cannot be accepted due to circumstances at the receiver site. The Initiator must cease transmission until it receives an RR packet which indicates that transmission may be resumed.

FORMAT	COMMANDS	RESPONSES	CONTROL WORD FORMAT							
			0	1	2	3	4	5	6	7
INFORMATION TRANSFER	I - INFORMATION		N(R)			P	N(S)			0
SUPERVISORY		RR - RECEIVER READY	N(R)			F	0	0	0	1
		RNR - RECEIVER NOT READY	N(R)			F	0	1	0	1
		REJ - REJECT	N(R)			F	1	0	0	1
UNNUMBERED	SARM - SET ASYNCRONOUS RESONSE MODE		0	0	0	P	1	1	1	1
	DISC - DISCONNECT		0	1	0	P	0	0	1	1
		UA - UNNUMBERED ACKNOWLEDGEMENT	0	1	1	F	0	0	1	1
		CMDR - COMMAND REJECT	1	0	0	F	0	1	1	1

PACKET CLASSIFICATIONS

AND

CONTROL WORD FORMATS

FIGURE 5

The REJ or Reject response packet is issued by the Responder process to inform the Initiator that an invalid data packet has been received. The Initiator process responds by transmitting the data packet again. If the Initiator had already sent packets after the erroneous one, they will also be retransmitted.

There are two types of Unnumbered Command packets. The SARM or Set Asynchronous Response Mode packet is issued by the Initiator process to bring the communication link to a logical 'up' state. Conversely, the DISC or Disconnect packet is issued by the Initiator process to force the communication link to a logical 'down' state. In both cases, the Responder process acknowledges receipt of these commands with an UA or Unnumbered Acknowledgement response packet.

A CMDR or Command Reject response packet is issued by the Responder process when it receives a command packet which cannot be recognized. This forces the Initiator process into the restart state, in which the Initiator attempts to bring the communication link to a logical 'up' state. This response is implemented but never used in this thesis. The CMDR packet is issued by the Responder process when it receives an unexpected command packet; other than an Information Transfer packet. This response should not be confused with a REJ response packet which is issued upon reception of an erroneous data packet.

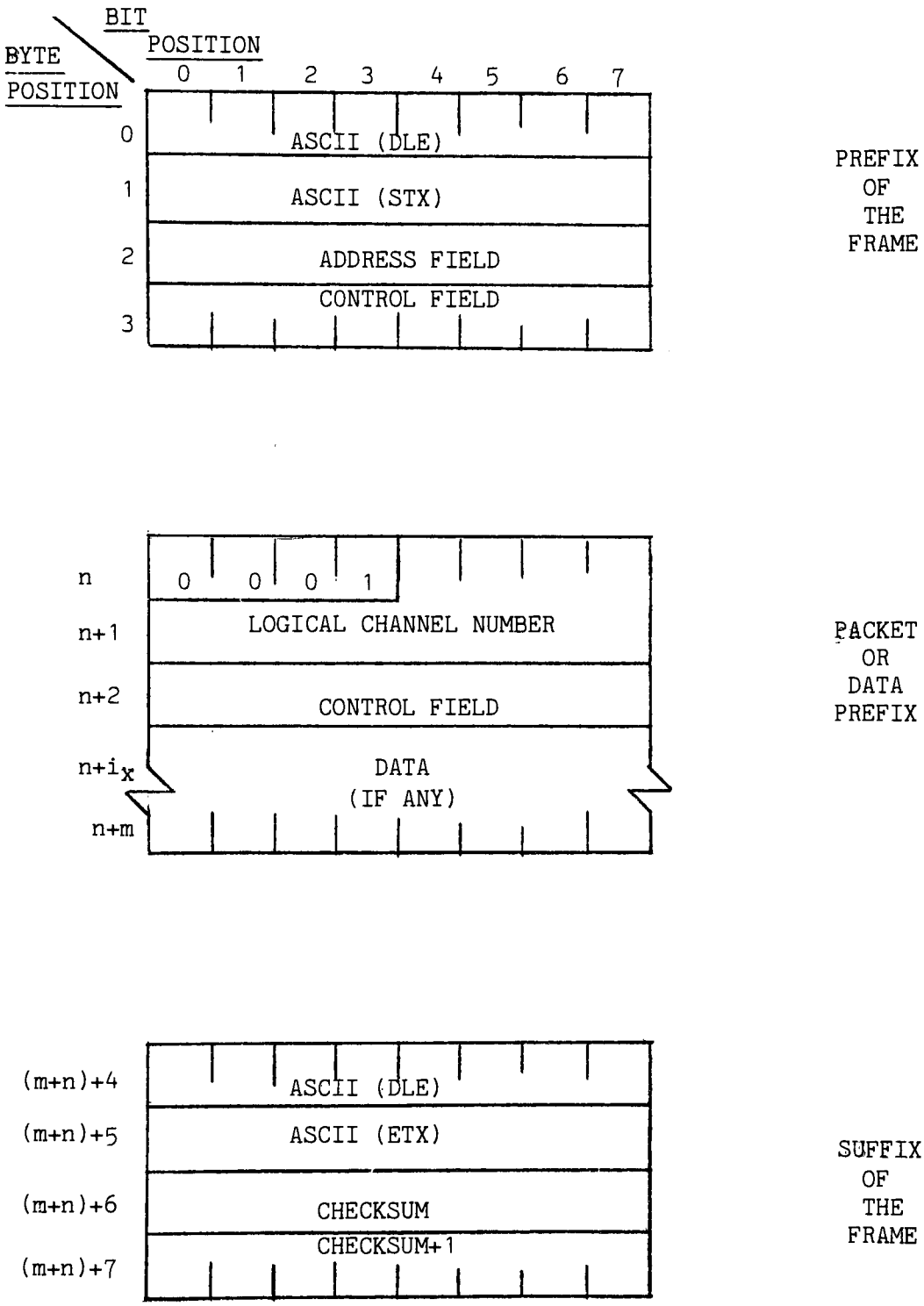
Framed Packets

Except for the data packet, each packet which is implemented in this thesis is three bytes in length. Data packets consist of a three byte header followed by the data to be sent. Figure 6 illustrates the packet structure.

The first byte consists of an identification code and the most significant bits of the logical channel number. The identifier code is located in the four most significant bit locations and is a binary one for all packets. The remaining four bits of the first byte concatenated with the eight bits of the second byte constitute the logical channel number which identifies the destination DTE communication node in a multiuser network. The third byte of the packet is known as the control field, and its format is individually tailored to the specific packet. This field identifies the packet type.

Each packet is framed before being transmitted. The frame consists of a four byte prefix before the packet and a four byte suffix after the packet, and is used to determine the validity of the packet at the receiving site. The first two bytes of both the prefix and suffix are command codes. The prefix code signifies the start of a frame, and the suffix code indicates the termination of a frame. The two bytes which follow the termination code are used for checksum purposes.

The third byte of the prefix, known as the address field, indicates the direction of the data flow. If the field contains a binary one, the packets originate at the DTE node site. If the field contains a binary



STRUCTURE OF A FRAMED PACKET

FIGURE 6

three, the packets originate at the DCE node site. Responses to the command packets that originate from the DTE node site have a binary representation of one; responses to those command packets originating from the DCE node site have a field value of three.

The last byte of the prefix is known as the control byte. It is identical to the control byte found in the packet except for bit position 3. The Poll/Final bit located in the Packet's control word represents the repeat status of the packet. If either the Command or Response packets have been transmitted more than once, the bit value will be set; normally, the status bit is zero. The More bit located in the frame's control word represents the continuation of data in the next data packet and is indicated by a logical one. If the entire message is contained within a data packet, the bit value will be zero. Figure 5 illustrates the format of each control word representing the packets which are implemented in this thesis.

The N(S) value in the control word of the data packet is the sequence number. It is incremented by one for each consecutive transmission of a data packet. The value of N(R) is the returned sequence number found within a RR, RNR, or REJ response packet. This field is one greater than the transmitted sequence number and found in the RR and RNR response packets. This number indicates the accepted data packets plus one. This means that all previous outstanding data packets whose value is less than, but not equal to, are affirmatively acknowledged. The field value found in the REJ response packet indicates the outstanding data packet which was found to be in error and therefore, must be retransmitted along with all of

the outstanding packets which follow it. The value represented within the control word of the data packet is considered the piggy-back option. The remaining bit positions of the control word represent the packet type.

Modifications

Because of the restrictions in this implementation, the entire packet header is redundant. The identifier and the logical channel number found in the first two bytes of the packet are not used in a two node system. The control byte appears in the frame prefix as well as the packet itself, and it too, is not mandatory in a two node system. Because of this redundancy, the packet header itself is not transmitted only the frame and any data to be transmitted are actually sent.

In addition, there are two different ways for transmitting an affirmative acknowledgement response. One possible approach is to transmit an RR packet. This is the method used exclusively by this thesis. The other approach is the piggy-back method. This technique, which Recommendation X.25 supports, allows for the reduction of transmitted packets. Assume the communication is occurring simultaneously between the DTE and DCE communication nodes at a rate producing 100% channel utilization. The number of packets which could be saved would be 50% using the rider or piggy-back option. For lower channel utilization the less efficient is the piggy-back option. Because of the implementation structure of this thesis, it becomes difficult but not impossible to implement this option.

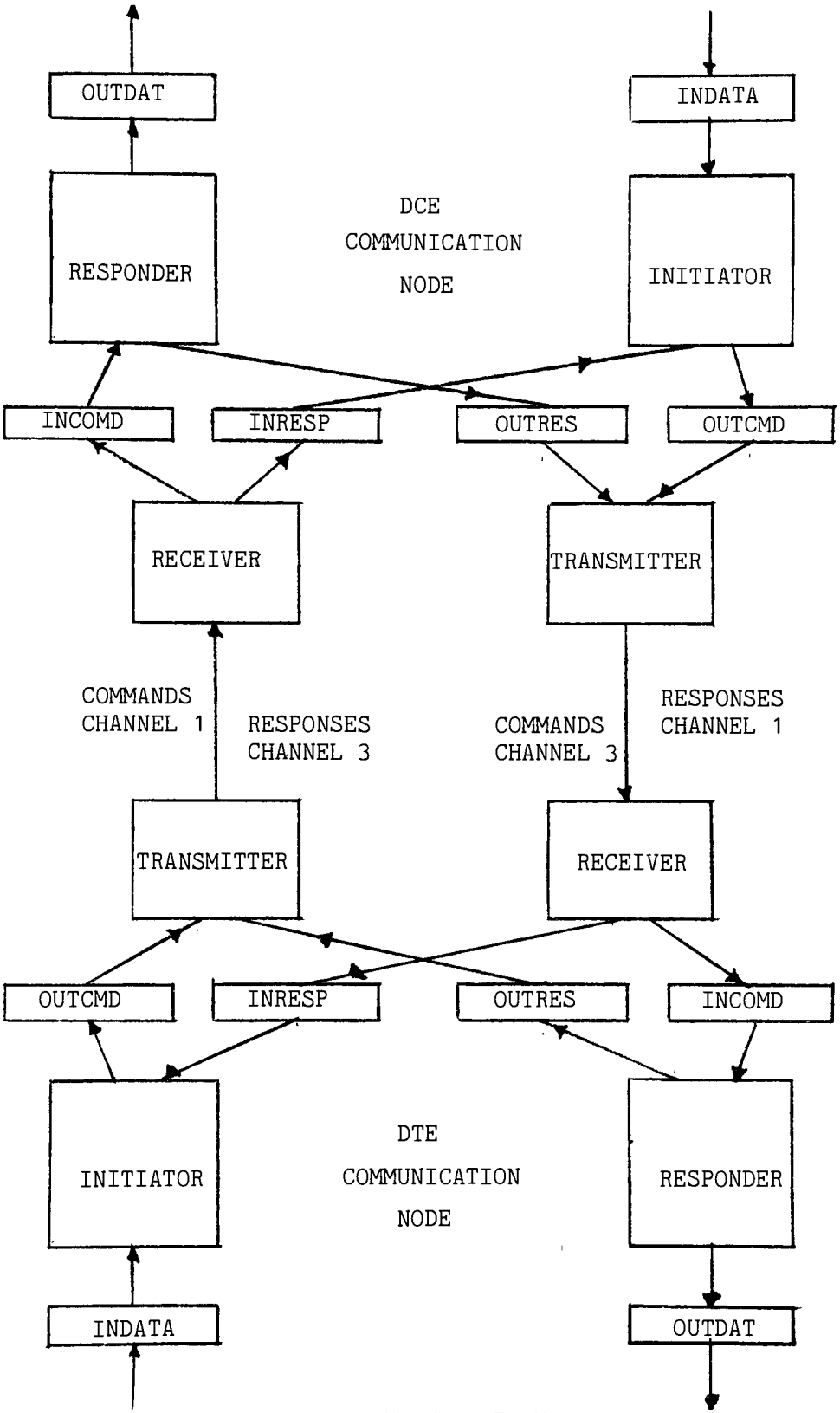
Finally, transparency bytes are inserted into the data portion of a packet when needed to prevent mistaking data for command sequences. They are removed by the Receiver process in order to reconstruct the original message. Transparency bytes are used in Recommendation X.25, because of the

ASCII restriction, only the control word would require the use of a transparency byte following it.

CHAPTER III

A SCENARIO OF A TRANSFER

This chapter covers the details of the transfer of information from a DTE to a DCE communication node from the creation of a packet to the disposal of the information packet after the receipt of its acknowledgement. The DTE communication node will be assumed to be the local site throughout this discussion. Because of the full duplex operation, transmissions originating at the DCE node are identical. Figure 7 illustrations in block mode the connection of two communication nodes and the internal communication paths of each node. The Reader may wish to refer to the earlier diagrams as well as the program listings while perusing this chapter.



SYSTEM COMMUNICATION

BLOCK DIAGRAM

FIGURE 7

PACKET CREATION

Packets of information are created within the Input process. If the character generator is not implemented, striking a key on the keyboard will generate an ASCII character which is transmitted to the Input process. An internal loop within the task collects the characters in a buffer node as they are generated until the desired character count is reached. The packet is passed to the Initiator process through the INDATA IPC buffer pair.

When the Input process has initialized the Input buffer of the buffer pair, it will validate the buffer by storing a nonzero value into its first byte. The second byte holds a count of the data. The third and fourth bytes in the IPC buffer constitute a sixteen bit field which holds the address of the node containing the data. Only the address of the node is passed from process to process, not the data itself. The format of this buffer may be reviewed in Figures 3 and 4.

After the transfer has been completed, the Input process fetches another node and enters the collection routine to wait for more characters to be entered from the keyboard. The next data packet will not be issued to the Initiator process until the previous packet has been accepted. If the Input process has to wait for the Initiator task to accept the previous packet, data from the keyboard may be lost. In that situation the Input process would not be accepting characters from the keyboard while it is waiting for the Input buffer to become invalid or empty.

ISSUING AN INFORMATION PACKET

When the Input buffer of the INDATA IPC buffer pair becomes valid, the two buffers of the buffer pair are exchanged. This action presents the Initiator process with information for the new data packet and gives the Input process an empty buffer for the next data. It is the responsibility of the Initiator process to transfer the data to the Transmitter process at the appropriate time for packet transmission.

The information which is contained in the Output buffer of the INDATA IPC buffer pair is queued into the Input table, which can hold up to eight packets at any one time. The contents of the Output buffer will not be queued into the table unless there exists at least two empty locations in the table, one for the contents of the Output buffer and the other to mark the end of the queue.

When the data is copied from the Input table for transmission, it is left in the table so as to keep a record of the outstanding packets which have not yet been acknowledged. When the Initiator process receives an affirmative acknowledgement as a response for one of its outstanding packets, the data sent in that packet and all previously transmitted but unacknowledged packets are discarded from the table and the node is returned to the node pool.

Figure 3 illustates the OUTCMD IPC buffer. The node address which contains the data packet is stored in the seventh and eighth bytes of the OUTCMD IPC buffer pair. The byte count which is also retrieved from the Input table is stored in the third byte during the dequeuing process. In

addition to this information, the sequence number of the packet is stored in the fifth location of the Input buffer. The value is derived from the table position from which the data came. The Poll/Final bit is added to the Input buffer.

The Initiator process finishes preparing the packets for transmission by storing the packet type code in the second location of the buffer, and validating it by storing a nonzero value into the first location of the buffer. The Transmitter process then accepts the information. When the Input buffer becomes available, the Initiator process may continue to transfer packets to the Transmitter task when there are seven outstanding packets or there is no data to send. The number of outstanding packets is governed by Recommendation X.25 and may vary between one and seven. The Initiator process must accept and process responses for its outstanding packets.

TRANSMITTING A PACKET

Once the Input buffer of the OUTCMD IPC buffer pair becomes valid, it is the responsibility of the Transmitter process to frame and transmit the packet of information. This process polls between its two inputs, the OUTCMD and the OUTRES IPC buffers. When there is a command packet to be transmitted, the process will exchange the buffers contained within the buffer pair, thus presenting to the Initiator process an empty or invalid buffer. The transmitter process begins the processing of the information of the command packet.

The Output buffer of the OUTRES IPC buffer pair is then scanned. If this buffer contains an outstanding RR response packet, it could be incorporated into the transmission of a data packet using the piggy-back option to reduce the number of packets that have to be transmitted. Since this option is not implemented in this thesis, an RR response packet is transmitted first, followed by the command packet.

The four byte prefix of the frame for the packet, shown in Figure 6 is transmitted first. As each byte is sent, it is added to the checksum. Following the transmission of the frame, the data is transmitted. The address of the node containing the data and the byte count are fetched from the Input table and the data is sent, byte by byte. The byte count is decremented as each byte is sent, and when it reaches zero, all of the data has been transmitted. Each data character is added to the checksum prior to being sent. If the data to be sent contains ASCII DLE characters, transparency bytes are inserted and added to the checksum.

The four byte suffix shouwn in Figure 5 is transmitted following the data. The DLE and ETX characters are added to the checksum before being transmitted. The final two characters of the suffix are the checksum. The two's complement of the checksum is computed, and the most significant byte is transmitted first. Once the checksum has been sent the Transmitter process will scan its input buffers waiting for a new packet to arrive.

RECEIVING A PACKET

The Receiver receives framed packets which were sent to it by its corresponding Transmitter process. Before a framed packet is received, a node from the node pool is fetched. Since the Receiver process does not know the length of the incoming frame, it assumes the largest possible value. The node pool is constructed such that each node contains 140 bytes to contain the entire frame.

All bytes received prior to a frame prefix are discarded. When the beginning of the frame is detected, the checksum is initialized and the Receiver prepares to save the data bytes. If a new frame header is discovered during reception, the partially collected packet is discarded and the Receiver begins from scratch with the new packet. If the Receiver detects the absence of a required transparency byte, the packet is also terminated. As the framed packet is received, the individual bytes are stored in the node. Every byte which is received is added to the accumulated checksum quantity. Transparency characters are added to the checksum and then discarded.

The Receiver process normally terminates the reception of a framed packet when the frame suffix is detected. The checksum contained in the suffix is added to the calculated checksum and if the result is zero the information in the packet is assumed to be correct.

The first two bytes of the acquired data node contain the address and control fields for the packet. The address field will contain either a binary one or three. This value informs the Receiver process of the

destination for the packet, and is easily processed due to the fact that the source code predetermines the destination. At the DCE communication node a received address of one indicates a command packet that should be sent to the Responder processs through the INCOMD IPC buffer pair. A received value of three indicates a response packet that should be sent to the Initiator process through the INRESP IPC buffer pair. If any other value is found in this location, the Receiver process will discard the data, since it cannot determine its proper destination.

The control word is then broken down into all four information fields, even through for some packet types, some of the fields could contain redundant information. This is not a concern since the destination process will only look at the pertinent fields.

After an information packet has been received, the address of the data node, the sequnce number, the Poll/Final status bit, and byte count are stored in the Input buffer of the INCOMD IPC buffer. The Receiver then validates the buffer, making it available to the Responder for processing.

If the piggy-back option is enabled, the Receiver process will insert into the Input buffer of the INRESP IPC buffer pair the data necessary to inform the Initiator process of the presence of a RR response packet. With the exception of the data packet, all nodes are returned to the node pool before another packet is received.

RESPONDING TO A COMMAND PACKET

In processing a received packet, the Responder first examines the packet type. If this value is negative, it is assumed that the Receiver task has detected an error during reception, usually a mismatch of the checksum. The Responder process will enter the Node Rejection state, issue a rejection response packet, and discard the data.

If the packet type field contains a legal value, the Final bit status maintained in the Responder process is compared with the Final bit in the packet. If the two values match, processing continues. A mismatch indicates that the Transmitter retransmitted a packet that was rejected, or that it did not retransmit a rejected packet. In either case, the Receiver responds with a Rejection packet.

The sequence number of the data packet is checked to determine whether the received data is in the correct sequence. This value must be one greater than the previously issued sequence number. Again, a rejection response packet will be issued if the packet is found to be out of sequence. If found to be valid, the data is queued into the Output table, which will hold it until the Output process is ready to receive it.

The selection of response packets for previously received data guarantees that there will be room in the table for the current packet. After queueing valid data into the table, if there is still room in the table, an RR response packet is sent. If the table is logically full, an RNR response is sent, which inhibits the transmitter process from sending any more packets.

The response packet is sent to the OUTRES IPC buffer pair. The returned sequence number and the status of the Final bit are stored, and the buffer is validated, making it available to the Transmitter process for transmission back to the DTE communication node.

When the Output process is ready for data, an address and data count are dequeued from the Output table and they are given to the Output process through the OUTDAT IPC buffer pair. The Responder process then validates the Input buffer, allowing the Output process to accept and dispose of the data.

The Responder process may accept and queue data packets presented to it by the Receiver task provided that the Responder process is in the Normal state and the Output table is not filled. Data packets are issued to the Output process on demand until the Output table is empty. This buffering tends to shield variations in time between packet arrivals from the processes that create or consume the data.

ISSUING DATA

The Output task receives the data packet from the Responder process through the OUTDAT IPC buffer pair by exchanging the buffers contained within the buffer pair. The data are fetched from the buffer in order, one character at a time, and fed to the CRT terminal where they are displayed. The characters are sent as fast as the bit rate will allow.

The byte count which was passed to the Output process along with the data packet address specifies the length of the data. The first two characters of the data packet are control and channel bytes, and are not part of the data itself, so the first data character is located in the third position within the packet. Similarly, the last two bytes in the packet following the data are the termination command bytes.

Once the byte count reaches zero and all of the information has been sent to the Output device, the node which contained the data packet is returned to the node pool.

The Output process is then ready to accept additional data nodes. If a bottleneck develops at the output device which delays the data to the terminal, no more data nodes will be accepted. Only after the last data character is transmitted to the terminal will the Output process accept another packet.

TRANSMITTING, RECEIVING & PROCESSING A RESPONSE PACKET

After queuing the data from a received packet for the Output process, the Responder process builds a response packet and passes it to the Transmitter. This response may not be the acknowledgement for the data which is currently being passed by the Output process due to the fact that the Output table can hold several packets of information.

The Transmitter process will accept the response packet from the Responder task through the OUTRES IPC buffer pair. The transmission of an affirmative acknowledgement RR response packet can be done in one of two ways. The piggy-back option allows a RR response to ride within the control word of a data packet. Otherwise, the response information is transmitted as a separate packet.

Response packets transmitted by the DCE communication node contain a channel identifier of one. The Receiver process of the DTE node uses it to determine the destination process for the packet, in this case, the Initiator task of the DTE communication node. The packet will be accepted by the Initiator process through the INRESP IPC buffer pair.

After accepting the packet, the Initiator checks the sequence number to determine whether the response for any of the outstanding data packets is contained in the Input table. If the sequence number does not match any of the packets, the Initiator process assumes the worst and begins restart procedures. On the other hand, if the sequence number is valid, the acknowledged data packet is dequeued from the Input table.

If the rejected response is received, the Initiator process enters the

Host Rejection state and retransmits the rejected data packet and all outstanding data packets which follow it. This action recovers from the error and returns the system to the Normal state.

CHAPTER IV

FURTHER DEVELOPMENT

Further development on this work could form the basis for future projects. There are may areas one could pursue; this section discusses one possibility.

The first subproject would be the implementation of code to bring the communication link from a logical 'down' state to a logical 'up' state, and would permit implementation of additonal packet types of Recommendation X.25.

Another subproject could be the implementation of a one node network. This could be conceived by incorporating the software for two DCE nodes into a central communication 'switch' and having two DTE nodes connected to the switch. This implies that a packet transmitted from a DTE node is acknowledged by its corresponding DCE network node before the data packet is transmitted to the destination DTE node from the other DCE node of the switch. This further implies that the receiving DCE node cannot acknowledge a received packet unless it knows the status of the destination DTE node.

Once the Calling packet has been implemented, the initiating DTE node will be able to communicate with selected DTE nodes connected to the central switch or one node network. This implies yet another subproject. However, the implementation of this structure is nontrivial, and would require the current transmission procedure to be modified.

Currently, the framed packet which is transmitted does not include most of the Recommendation X.25 packet strucuture. The full implementation would require that the Transmitter process receive two additional fields through its IPC buffers: logical address and data verifier. The logical address would identify the destination DTE of the user in a multi-node network. The data verifier indicates the continuation of the data, and is cleared on the last packet in a transmission.

In order to initiate the Calling packet, the user must be able to issue commands to the Input process. The RITLP protocol is suggested as one possible way to implement these changes. RITLP was not implemented earlier in order to keep the communication between the user and DTE node as simple as possible for this first project. We could demonstrate our project using a simple ASCII terminal. These added features would implement Recommendation X.25 on the scale which was originally proposed.

It is easily seen that the continuance of this thesis is not a trivial matter. However, the present architecture provides a solid foundation for future work.

REFERENCES

1) Standard Network Protocol for the DATAPAC Network;
Ottawa, Canada
Trans-Canada Telephone System, 1977, p. 82.

2) Telenet X.25 Documentation Service; U.S.A.
Telenet Communication Corporation, 1977, p. 187.

3) M6800 Microcomputer System Data Book; U.S.A.
Motorola Incorporated, 1976, p. 165.

APPENDIX

1) Assembled source listing for
the Recommendation X.25 Protocol.

2) Assembled source listing for
the Multitasking Monitor System.

```
DDDD1                 ************************************************************************
DDDD2                 ************************************************************************
DDDD3                 **                                                                    **
DDDD4                 **                                                                    **
DDDD5                 **    RECDMMENDATIDN X.25 PRDTDCDL                                    **
DDDD6                 **                                                                    **
DDDD7                 **    RECDMMENDATIDN X.25 IS A DATA PACKET TRANSMISSIDN AND           **
DDDD8                 **    RECEPTIDN PRDTDCDL DESIGNED AND ADDPTED IN 1976 BY THE          **
DDDD9                 **    INTERNATIDNAL TELEPHDNE AND TELEGRAPH CDMMUNICATIDN             **
DDD1D                 **    CDMMITTEE (C.C.I.T.T.).                                         **
DDD11                 **    THERE ARE SEVEN PRDCESSES WHICH CDNSTITUTE THIS PRDGRAM.        **
DDD12                 **    THEY ARE THE INITIATDR, RESPDNDER, TRANSMITTER, RECEIVER,       **
DDD13                 **    INPUT, DUTPUT, AND TIMER TASKS.  THIS PRDGRAM REPRESENTS        **
DDD14                 **    BDTH SIDES DF THE CDMMUNICATIDN LINK.  WHEN THE VARIABLE        **
DDD15                 **    (CHANGE) IS ZERD, THE (DTE) IS REPRESENTED.  WHEN THE VALUE     **
DDD16                 **    IS NDN-ZERD, THE (DCE) IS ASSUMED.  THE (DTE) CDNSTITUTES       **
DDD17                 **    PART DF THE HDST CDMPUTER WHILE THE (DCE) IS CDNSIDERED         **
DDD18                 **    THE INTERFACE INTD THE NETWDRK.                                 **
DDD19                 **                                                                    **
DDD2D                 **    PACKET INFDRMATIDN IS TRANSMITTED AND RECEIVED BETWEEN          **
DDD21                 **    PRDCESSES VIA THE (INTFAC) TABLE.  INPUT AND DUPUT PACKET       **
DDD22                 **    INFDRMATIDN IS STDRED IN THE QUEUES (INTA8L), AND (DUTA8L),     **
DDD23                 **    RESPECTIVELY.  DATA CDNTAINED WITHIN THE PACKETS ARE STDRED     **
DDD24                 **    IN VARIABLE SIZES NDT TD EXCEED 128 BYTES.  THE STDRAGE         **
DDD25                 **    AREA IS DETERMINED FRDM THE RDUTINE (GETNDD) WHICH RETRIEVES    **
DDD26                 **    A FIXED NDDE SIZE FRDM THE DATA NDDE PDDL.                      **
DDD27                 **                                                                    **
DDD28                 **                                                                    **
DDD29                 ************************************************************************
DDD3D                 ************************************************************************

DDD32                 ************************************************************************
DDD33                 ************************************************************************
DDD34                 **                                                                    **
DDD35                 **                                                                    **
DDD36                 **    WRITTEN BY:      LDUIS D. NYERGES                               **
DDD37                 **                     CANDIDATE FDR MASTERS DEGREE                   **
DDD38                 **                     CDMPUTER ENGINEERING                           **
DDD39                 **                     RDY S. CZERNIKDWSKI, ADVISDR                   **
DDD4D                 **                                                                    **
DDD41                 **    SYSTEM:          MDTDRDLA EXDRCISER;    M68DD BASED             **
DDD42                 **    MEMDRY                                                          **
DDD43                 **      REQUIREMENTS: 32K RAM                                         **
DDD44                 **    HARDWARE                                                        **
DDD45                 **      REQUIREMENTS: I/D BDARD CDNSISTING DF FDUR ACIA'S             **
DDD46                 **                    REAL TIME CLDCK DF 2152 HERTZ                   **
DDD47                 **    SDFTWARE                                                        **
DDD48                 **      REQUIREMENTS: MULTI-TASKING MDNITDR SYSTEM                    **
DDD49                 **    ADDITIDNAL                                                      **
DDD5D                 **      INFDRMATIDN:  PLEASE REFERENCE THESIS                         **
DDD51                 **                    "AN INVESTIGATIDN AND IMPLEMENTATIDN            **
DDD52                 **                     DF THE C.C.I.T.T. RECDMMENDATIDN X.25          **
DDD53                 **                     DN A M68DD PILDT BASED NETWDRK"                **
DDD54                 **                                                                    **
DDD55                 **                                                                    **
DDD56                 ************************************************************************
DDD57                 ************************************************************************
```

```
00059                 *  ASSEMBLY DIRECTIVES


00061                 *  OPT - (OPTIONS)

00063                 *    REL  - RELOCATABLE OBJECT CODE
00064                 *    OBJ  - BUILD OBJECT FILE
00065                 *    PAGE - LINES PER PAGE
00066                 *    LLEN - LINE LENGTH
00067                 *    MEX  - PRINT MACRO EXPANSIONS
00068                 *    MD   - LIST MACRO DEFINITIONS
00069                 *    MC   - LIST MACRO CALLS
00070                 *    GEN  - LIST MULTIPLE LINES OF FCC, FCB, AND FDB
00071                 *    UNA  - LIST NON-ASSEMBLED CODE
00072                 *    CLIST- PRINT CONDITIONAL DIRECTIVES
00073                 *  TTL  - PRINT TITLE AT TOP OF EACH PAGE
00074                 *  IDNT - IDENTIFICATION LABEL FOR OBJECT FILE
00075                 *  XDEF - EXTERNAL DEFINITIONS
00076                 *            (LABELS ARE DEFINED HERE)
00077                 *  XREF   EXTERNAL REFERENCES
00078                 *            (LABELS ARE DEFINED ELSEWHERE)



00080                       NAM    THESIS:X.25
00081                       OPT    REL,OBJ
00082                       OPT    PAGE=58,LLEN=96
00083                       OPT    MEX,MD,MC,GEN,UNA,CLIST
00084                       TTL    *** RECOMMENDATION X.25 PROTOCOL ***
00085                       IDNT   *** L.D. NYERGES - FALL, '80   X.25

00087          0000  A CHANGE SET    0        VALUE IS ZERO FOR (DTE) AND NON-ZERO FOR (DCE)

00089          0000  A        IFEQ   CHANGE   ASSEMBLE THIS CODE IF VALUE IS ZERO
00090                       XDEF   TKPC0,TKPC1,TKPC2,TKPC3,TKPC4
00091                       XDEF   TKPC5,TKPC6,TKPCM,INTDTE
00092                       ENDC


00094          0000  A        IFNE   CHANGE   ASSEMBLE THIS CODE IF VALUE IS NON-ZERO
00095                    XDEF TKPC7,TKPC8,TKPC9,TKPCA
00096                    XDEF TKPC8,TKPCC,TKPCD,INTDCE
00097                    ENDC


00099                       XREF   PSCT:GETCHR,PUTCHR,XCHSTS
00100                       XREF   DSCT:CRTFLG
```

```
00102                   *  ABSOLUTE ADDRESS


00104                   *  THE FOLLOWING LABELS TAKE THE ASSIGNED VALUES AT ASSEMBLY
00105                   *  TIME.  THEY GOVERN THE I/O PORT OEFINITIONS, BUFFER SIZES,
00106                   *  ANO TIMER OELAYS.

00108                   *  NOTE:  CHANGES TO ANY OF THESE VALUES MAY IMPLY A
00109                   *         OATA STRUCTURE CHANGE AND THEREFORE, RECODING
00110                   *         OF TABLES IS ASSUMED.


00112A 0000                     ASCT

00114           0000  A NETWRK EQU    0        PORT NUMBER FOR HOST   (DTE) TO (DCE)
00115           0001  A SUBNET EQU    1        PORT NUMBER FOR NETWORK   (DCE) TO (DTE)
00116           0002  A URACIA EQU    2        USER PORT NUMBER - USER TO (DTE)
00117           0003  A SBACIA EQU    3        NETWORK PORT NUMBER   NETWORK TO (DCE)
00118           0007  A KCONST EQU    7        MAXIMUM NUMBER OF OUTSTANDING PACKETS
00119           0004  A N2NUM  EQU    04       MAXIMUM REPEAT COUNT FOR TRANSMISSION RETRY
00120           3000  A TIMCNT EQU    $3000    TIMEOUT DELAY COUNT
00121           008C  A NODESZ EQU    140      BUFFER PACKET NODE SIZE
00122           0028  A NODESN EQU    40       MAXIMUM NUMBER OF NODES IN POOL
00123           0008  A IPCBFZ EQU    8        SIZE OF INTERPROCESS COMMUNICATION BUFFERS
```

```
00125                   *  DATA SECTION


00127                   *  PERMANENT AND TEMPORARY LOCATIONS FOR ALL SEVEN PROCESSES ARE
0012B                   *  REPRESENTED BELOW.  AT INITIALIZATION, THIS ENTIRE TABLE IS
00129                   *  ZERO'ED OUT.  THE TABLES (NODTBL), (INTFAC) AND (BUFPNT)
00130                   *  ARE INITIALIZED THROUGH THE (XXXINT) ROUTINES.


00132D 0000                     DSCT

00134           0000  D VARLST EQU   *        START OF DATA AREA
00135D 0000     0002  A BUFNOD RMB   2        POINTER TO NEXT DATA PACKET NODE
00136D 0002     15ED  A NODTBL RMB   NODESZ*NODESN NODE POOL
00137D 15E2     014D  A BUFPNT RMB   NODESN*IPCBFZ (IPC) PRIMARY AND SECONDARY BUFFER POINTERS
00138D 1722     001B  A INTABL RMB   3*8      INPUT BUFFER QUEUE
00139D 173A     001B  A OUTABL RMB   3*8      OUTPUT BUFFER QUEUE


00141           1752  D INTFAC EQU   *        PRIMARY AND SECONDARY IPC BUFFERS
00142D 1752     15E2  D INDATA FDB   BUFPNT   PRIMARY - (INPUT) TO (INITIATOR)
00143D 1754     15EA  D STOBY1 FDB   BUFPNT+IPCBFZ SECONDARY - (INPUT) TO (INITIATOR)

00145D 1756     15F2  D OUTCMD FDB   BUFPNT+(IPCBFZ*2) PRIMARY - (INITIATOR) TO (TRANSMITTER)
00146D 175B     15FA  D STOBY2 FDB   BUFPNT+(IPCBFZ*3) SECONDARY   (INITIATOR) TO (TRANSMITTER)

00148D 175A     1602  D INRESP FDB   BUFPNT+(IPCBFZ*4) PRIMARY   (RECEIVER) TO (RESPONDER)
00149D 175C     160A  D STOBY3 FDB   BUFPNT+(IPCBFZ*5) SECONDARY   (RECEIVER) TO (RESPONDER)

00151D 175E     1612  D OUTRES FDB   BUFPNT+(IPCBFZ*6) PRIMARY - (RESPONDER) TO (TRANSMITTER)
00152D 1760     161A  D STOBY4 FDB   BUFPNT+(IPCBFZ*7) SECONDARY   (RESPONDER) TO (TRANSMITTER)

00154D 1762     1622  D INCOMD FDB   BUFPNT+(IPCBFZ*8) PRIMARY - (RECEIVER) TO (INITIATOR)
00155D 1764     162A  D STOBY5 FDB   BUFPNT+(IPCBFZ*9) SECONDARY   (RECEIVER) TO (INITIATOR)

00157D 1766     1632  D OUTDAT FDB   BUFPNT+(IPCBFZ*10) PRIMARY   (RESPONDER) TO (OUTPUT)
00158D 176B     163A  D STOBY6 FDB   BUFPNT+(IPCBFZ*11) SECONDARY - (RESPONDER) TO (OUTPUT)
```

```
00160                   *  DYNAMIC POINTERS AND VARIABLES


00162          176A  D VARABL EQU    *

00164D 176A    0002  A CPTIM1 RMB    2        TIME REAMINING ON COUNTER
00165D 176C    0002  A TEMPR1 RMB    2        TEMPORARY LOCATION FOR (INPUT) USE
00166D 176E    0002  A TEMPR2 RMB    2        TEMPORARY LOCATION FOR (INITIATOR) USE
00167D 1770    0002  A TEMPR3 RMB    2        TEMPORARY LOCATION FOR (TRANSMITTER) USE
00168D 1772    0002  A TEMPR4 RMB    2        TEMPORARY LOCATION FOR (RECEIVER) USE

00170D 1774    0002  A TEMPR5 RMB    2        TEMPORARY LOCATION FOR (RESPONDER) USE
00171D 1776    0002  A TEMPR7 RMB    2        TEMPORARY LOCATION FOR (OUTPUT) USE
00172D 1778    0002  A TEMPX  RMB    2        TEMPORARY LOCATION FOR INITIALIZATION USE
00173D 177A    0002  A XMTMP  RMB    2        ADDRESS OF TRANSMITTING DATA

00175D 177C    0002  A INADDR RMB    2        INPUT NODE ADDRESS
00176D 177E    0002  A AVAILN RMB    2        RECEIVER INPUT NODE ADDRESS
00177D 1780    0002  A RCSUM  RMB    2        RECEIVER CHECKSUM
00178D 1782    0002  A OUTADR RMB    2        OUTPUT NODE ADDRESS
00179D 1784    0002  A XTCHCK RMB    2        TRANSMITTER CHECKSUM

00181D 1786    0001  A RNRVAL RMB    1        SEQUENCE NUMBER OF LAST RECEIVE NOT READY
00182D 1787    0001  A REJVAL RMB    1        SEQUENCE NUMBER OF LAST REJECT
00183D 1788    0001  A NBUSY  RMB    1        NODE BUSY INDICATOR
00184D 1789    0001  A NREJ   RMB    1        NODE REJECT INDICATOR
00185D 178A    0001  A HBUSY  RMB    1        HOST BUSY INDICATOR
```

```
001870 178B    0001  A HREJ   RMB    1        HOST REJECT INOICATOR
001880 178C    0001  A VOFS   RMB    1        NEXT SEQUENCE NUMBER TO TRANSMIT
001890 178O    0001  A VOFR   RMB    1        EXPECTEO NEXT SEQUENCE NUMBER RECEIVEO
001900 178E    0001  A NPROFR RMB    1        EXPECTEO RECEIVEO ACKNOWLEOGEO SEQUENCE NUMBER
001910 17BF    0001  A NT1OFR RMB    1        RECEIVING SEQUENCE NUM8ER

001930 179O    0001  A NPROFS RMB    1        RECEIVEO SEQUENCE NUMBER
001940 1791    0001  A NEXT   RMB    1        REAR POINTER OF INPUT QUEUE
001950 1792    0001  A VALIOF RMB    1        FRONT POINTER OF OUTPUT QUEUE
001960 1793    0001  A RCOUNT RMB    1        REPEAT COUNTER
001970 1794    0001  A TBUSY  RMB    1        TIMEOUT INOICATOR

001990 1795    0001  A VALIOR RMB    1        REAR POINTER FOR OUTPUT QUEUE
002000 1796    0001  A HCBUSY RMB    1        SWITCH FOR RECEIVER RESTART
002010 1797    0001  A NOFR   RMB    1        TRANSMITTING SEQUENCE NUMBER
002020 1798    0001  A RRFLG  RMB    1        SWITCH FOR BUFFER SELECTION FOR TRANSMITTER

002040 1799    0001  A FINBIT RMB    1        FINAL BIT TO BE TRANSMITTEO
002050 179A    0001  A POLLBT RMB    1        POLL BIT TO BE RECEIVEO
002060 179B    0001  A POLFIN RMB    1        RECEIVING POLL/FINAL BIT
002070 179C    0001  A INCNT  RMB    1        INPUT BYTE COUNTER

002090 179O    0001  A OUTCNT RMB    1        OUTPUT BYTE COUNTER
002100 179E    0001  A CHANNL RMB    1        RECEIVEO CHANNEL NUMBER
002110 179F    0001  A RCBYTE RMB    1        RECEIVING BYTE COUNT
002120 17AO    0001  A XTBYTE RMB    1        TRANSMITTING BYTE COUNT
002130 17A1    0001  A HONLNX RMB    1        HOST TRANSMITTER ON LINE

002150 17A2    0001  A HONLNR RMB    1        HOST RECEIVER ON LINE
002160 17A3    0001  A XTCNTL RM8    1        TRANSMITTER CONTROL WORO
002170 17A4    0001  A XMTYPE RMB    1        TRANSMIT PACKET TYPE
002180 17A5    0001  A TIME1  RMB    1        TIMEOUT INOICATOR
002190 17A6    0001  A XVAR   RMB    1        LAST ACKNOWLEOGEO SEQUENCE NUMBER

002210 17A7    0001  A OATA   RMB    1        TEMPORARY HOLOING LOCATION FOR OUTPUT
002220 17AB    0001  A RCLIMT RMB    1        RECEIVING OATA PACKET LIMIT
00223          17A9  O ENOVAR EQU    *
```

```
00225                   ************************************************************************
00226                   *                                                                      *
00227                   *    INITIALIZATION AND ROUTINES                                       *
00228                   *                                                                      *
00229                   *    THIS SECTION CONTAINS SIX ROUTINES.  THEY ARE THE INITIALIZER,    *
00230                   *    GETNOD, PUTNOD, START1, RESET1, AND WAITBF.  GETNOD, PUTNOD       *
00231                   *    AND WAITBF ARE USED BY MORE THAN ONE TASK.  THEY ARE THEREFORE,   *
00232                   *    RE-ENTRIENT.  THE INITIALIZER INITIALIZES THE BUFFERS TO THEIR    *
00233                   *    CORRECT VALUES BEFORE X.25 BEGINS EXECUTION.  THE MONITOR         *
00234                   *    INVOKES THE INITIALIZER.  GETNOD AND PUTNOD REMOVES AND RETURNS   *
00235                   *    NODES TO THE NODE POOL, RESPECTIVELY.  WAITBF IS A ROUTINE        *
00236                   *    THAT WILL WAIT FOR A BUFFER TO BECOME INVALID.  START AND RESET1  *
00237                   *    CONTROL THE TIMER START/STOP RELATIONSHIP, RESPECTIVELY.          *
00238                   *                                                                      *
00239                   ************************************************************************



00241                   *  INITIALIZER

00243                   *  THERE ARE THREE FUNCTIONS IN INITIALIZING FOR X.25.  THE FIRST
00244                   *  CLEARS OUT THE ENTIRE BUFFER SECTION.  THE SECOND FUNCTION
00245                   *  INITIALIZES THE TABLE (NODTBL).  THE FINAL FUNCTION IS TO
00246                   *  INSERT THE PRIMARY AND SECONDARY POINTERS INTO THE TABLE (BUFPNT).


0024BP 0000                       PSCT

00250          0000  A            IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
00251          0000  P INTDTE     EQU    *
00252                             ENDC

00254          0000  A            IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
00255                  INTDCE     EQU *
00256                    ENDC


00258                   *  CLEAR BUFFER

00260P 0000 CE 0000  D            LDX    #VARLST  FETCH STARTING ADDRESS
00261P 0003 6F 00    A VARCLR     CLR    0,X      CLEAR LOCATION
00262P 0005 08                    INX             ADVANCE BUFFER POINTER
00263P 0006 BC 17A9  D            CPX    #ENDVAR  CHECK FOR END OF BUFFER
00264P 0009 26 FB 0003            BNE    VARCLR   TEST IS POSITIVE IF VALUE IS ZERO
```

```
00266                    *  INITIALIZE (NODT8L)

00268                    *  THE NODE POOL IS INITIALIZED AS A FORWARD LINK LIST.  THE
00269                    *  POINTER (8UFNOD) POINTS TO THE NEXT AVAILA8LE NODE IN THE
00270                    *  POOL.  THE FIRST TWO 8YTES OF EACH NODE POINT TO THE NEXT
00271                    *  NOOE IN THE CHAIN.  THE LAST NODE WILL HAVE $0000 AS ITS
00272                    *  POINTER INDICATING END OF NODE.


00274P 0008 CE 0002  0          LDX    #NODT8L  FETCH INITIAL NODE
00275P 000E FF 0000  D          STX    8UFNOD   INITIALIZE NODE POINTER
00276P 0011 FF 1778  D          STX    TEMPX    MOVE INOEX ADDRESS INTO ACCUMULATORS
00277P 0014 86 1779  D          LDAA   TEMPX+1
00278P 0017 F6 1778  D          LDA8   TEMPX

00280P 001A 88 8C    A X25LP1   ADDA   #NODESZ  ADD TO ACCUMULATORS THE NODE SIZE
00281P 001C C9 00    A          ADC8   #0
00282P 001E A7 01    A          STAA   1,X       INITIALIZE NEXT NODE
00283P 0020 E7 00    A          STA8   0,X
00284P 0022 87 1779  D          STAA   TEMPX+1  RETURN NEW VALUE 8ACK TO INDEX REGISTER
00285P 0025 F7 1778  D          STA8   TEMPX
00286P 0028 FE 1778  D          LDX    TEMPX
00287P 0028 8C 1556  D          CPX    #NODT8L+(NODESZ*(NODESN-1)) CHECK FOR END OF NODE POOL
00288P 002E 26 EA 001A          8NE    X25LP1   TEST IS POSITIVE IF VALUE IS ZERO
```

```
00290                   *  INITIALIZE PRIMARY AND SECONDARY POINTERS

00292                   *  THE ADDRESSES OF THE PRIMARY AND SECONDARY BUFFERS OF (INTFAC)
00293                   *  ARE PASSED TO THE TABLE (BUFPNT) FOR INITIALIZATION.


00295P 0030 CE 15E2  D          LDX    #BUFPNT  FETCH DESTINATION POINTER
00296P 0033 FF 177B  D          STX    TEMPX
00297P 0036 CE 1752  D          LDX    #INTFAC  FETCH SOURCE POINTER
00298P 0039 B6 1779  D          LDAA   TEMPX+1
00299P 003C F6 177B  D          LDAB   TEMPX
00300P 003F A7 01    A X25LP3   STAA   1,X      INITIALIZE DESTINATION BUFFER
00301P 0041 E7 00    A          STAB   0,X
00302P 0043 8B 0B    A          ADDA   #IPCBFZ  INCREASE SOURCE POINTER BY (IPCBFZ)
00303P 0045 C9 00    A          ADCB   #0
00304P 0047 08                  INX             ADVANCE DESTINATION POINTER
00305P 004B 08                  INX
00306P 0049 8C 176A  D          CPX    #VARABL  CHECK FOR END OF TABLE
00307P 004C 26 F1 003F          BNE    X25LP3   TEST IS POSITIVE IF VALUE IS ZERO



00309                   *  SET (DATA) FOR OUTPUT VALUE OF AN ASCII SPACE (HEX 20).
00310                   *  THIS IS THE INITIAL VALUE FOR THE INPUT ROUTINE.  NORMALLY,
00311                   *  THE INPUT ROUTINE DOES NOT EXIST AND THEREFORE, THE
00312                   *  VARIABLE (DATA) WOULD NOT EXIST.

00314P 004E 86 20    A          LDAA   #$20
00315P 0050 B7 17A7  D          STAA   DATA



00317                   *  IT IS ASSUMED THAT THE CHANNELS ARE UP AND RUNNING.  THEREFORE,
00318                   *  THE VARIABLE (HONLNR) IS SET.  HONLNR IMPLIES THAT THE
00319                   *  HOST RECEIVER IS ON LINE IF SET TO ONE.

00321P 0053 7C 17A2  D          INC    HONLNR   HOST ON LINE RECEIVER LINK INDICATOR
00322P 0056 39                  RTS             RETURN TO MONITOR
```

```
00324                   *  FETCH A NODE

00326                   *  GET A NODE FROM THE NODE POOL.  THE VARIABLE (BUFNOD) POINTS
00327                   *  TO THE NEXT AVAILABLE NODE IN THE POOL.  IF ALL NODES ARE
00328                   *  EXTINGUISHED, THE ROUTINE WILL LOOP UNTIL A NODE HAS BEEN
00329                   *  RETURNED.  THIS IS A DYNAMIC LOOP WHICH RETURNS CONTROL
00330                   *  TO THE SCHEDULER FOR EACH CYCLE OF THE LOOP.  THIS IS A
00331                   *  CRITICAL REGION SINCE ITS COUNTER PART MAY RETURN
00332                   *  A NODE AT ANY TIME.


00334          0057  P GETNOD EQU    *
00335P 0057 0F                 SEI             DECLARE A CRITICAL REGION
00336P 005B 7C 0000  A NODAGN INC    CRTFLG    SET FOR CRITICAL REGION FOR EACH CYCLE
00337P 005B FE 0000  D        LDX    BUFNOD    FETCH ADDRESS OF NEXT NODE
00338P 005E 26 06 0066        BNE    GNODE     IF VALUE IS ZERO, NODES EXHAUSTED

00340                   *  NODE NOT FOUND.  RETURN CONTROL TO SCHEDULER
00341                   *  AND WAIT FOR NEXT SCHEDULED PASS.  CHECK AGAIN.

00343P 0060 7F 0000  A        CLR    CRTFLG    CLEAR CRITICAL REGION
00344P 0063 3F                SWI             RETURN CONTROL TO SCHEDULER
00345P 0064 20 F2 005B        BRA    NODAGN    REPEAT THE CYCLE

00347                   *  NODE HAS BEEN FOUND.  REPLACE (BUFNOD) WITH THE POINTER
00348                   *  CONTAINED IN THE NODE WHICH (BUFNOD) IS POINTING TO.

00350P 0066 FF 177B  D GNODE  STX    TEMPX     SAVE FOR FUTURE USE
00351P 0069 EE 00    A        LDX    0,X       FETCH NEXT NODE POINTER
00352P 006B FF 0000  D        STX    BUFNOD    UPDATE NODE POINTER
00353P 006E FE 177B  D        LDX    TEMPX     RETURN THE ORIGINAL NODE POINTER
00354P 0071 7A 0000  A        DEC    CRTFLG    NEXT CRITICAL REGION
00355P 0074 0E                CLI
00356P 0075 39                RTS
```

```
00358                   *  RETURN A NODE

00360                   *  PUT A NODE BACK INTO THE NODE POOL.  A NODE HAS BEEN USED
00361                   *  AND MUST BE RETURNED.  IT IS RETURNED AT THE FRONT OF THE
00362                   *  CHAIN OF NODES, (F.I.F.O.).  SINCE A NODE MAY BE REMOVED
00363                   *  AT ANY TIME, (PUTNOD) WILL OPERATE UNDER  A CRITICAL
00364                   *  REGION ENVIRONMENT.  ACCUMULATOR A AND B IS RESTORED.


00366          0076  P PUTNOD EQU    *
00367P 0076 0F                SEI             SET FOR A CRITICAL REGION
00368P 0077 7C 0000   A       INC     CRTFLG
00369P 007A 36                PSHA

00371                   *  THE NODE POINTER (BUFNOD) IS PLACE IN THE RETURNING
00372                   *  NODE AND THE POINTER TO THE RETURNING NODE IS PLACED
00373                   *  IN THE NODE POINTER (BUFNOD).

00375P 007B B6 0001   0       LOAA    BUFNOD+1 FETCH LSB FROM NODE POINTER
00376P 007E A7 01     A       STAA    1,X      UPDATE LSB OF THE RETURNING NODE POINTER
00377P 0080 B6 0000   0       LOAA    BUFNOD   FETCH THE MSB OF THE NODE POINTER
00378P 0083 A7 00     A       STAA    0,X      UPDATE THE RETURNING NODE POINTER, MSB
00379P 0085 32                PULA
00380P 0086 FF 0000   0       STX     BUFNOD   UPDATE NODE POINTER WITH RETURNING NODE
00381P 0089 7A 0000   A       DEC     CRTFLG   EXIT FROM CRITICAL REGION
00382P 008C 0E                CLI
00383P 008D 39                RTS
```

```
00385                    *  START TIMER

00387                    *  RECOMMENDATION REQUIRES A TIMER TO DETERMINE WHETHER A PACKET
00388                    *  OF INFORMATION ARRIVED AT ITS DESTINATION.  WHEN A PACKET IS
00389                    *  TRANSMITTED, THIS ROUTINE IS CALLED TO START THE TIMER.  THIS
00390                    *  ROUTINE MAY BE CALLED WHILE THE TIMER IS UNDER OPERATION.
00391                    *  IT, THEREFORE, WILL RESET THE TIMING VALUE AND START AGAIN.
00392                    *  (START1) WILL READY THE TIMER PROCESS.  ACCUMULATORS ARE
00393                    *  SAVED.  THIS ROUTINE OPERATES UNDER A CRITICAL REGION.


00395           008E  P START1 EQU    *
00396P 008E 37                 PSHB
00397P 008F 0F                 SEI             SET FOR CRITICAL REGION
00398P 0090 7C 0000  A         INC    CRTFLG
00399P 0093 7F 17A5  D         CLR    TIME1    CLEAR TIMEOUT FLAG
00400P 0096 CE 3000  A         LDX    #TIMCNT  FETCH TIMER TIME VALUE
00401P 0099 FF 176A  D         STX    CPTIM1   RESET TIMING VALUE

00403                    *  READY THE TIMER TASK.  ACCUMULATOR A CONTAINS THE
00404                    *  NEW STATUS WORD AND ACCUMULATOR B WILL HOLD THE TASK
00405                    *  NUMBER.  FOR THE (DTE) THE TASK IS 06 AND FOR THE
00406                    *  (DCE) THE TASK IS (0D HEX).

00408P 009C C6 00    A         LDAB   #$00     READY STATUS WORD
00409           0000  A         IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
00410P 009E 86 06    A         LDAA   #$06     (DTE) TIMER TASK NUMBER
00411                          ENDC

00413           0000  A         IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
00414                     LDA A #$0D  (DCE) TASK NUMBER
00415                     ENDC

00417P 00A0 8D 0000  A         JSR    XCHSTS   CHECK STATUS WORD ROUTINE
00418P 00A3 7A 0000  A         DEC    CRTFLG   EXIT FROM CRITICAL REGION
00419P 00A6 33                 PULB
00420P 00A7 39                 RTS
```

```
00422                   *  RESET TIMER

00424                   *  THIS ROUTINE WILL STOP THE TIMER FROM CONTINUING.  IT
00425                   *  WILL REPLACE THE STATUS WORD WITH A (HEX 80).  IT OPERATES
00426                   *  UNDER A CRITICAL REGION.  THE ACCUMULATORS ARE RESTORED.


00428           00A8  P RESET1 EQU    *
00429P 00A8 37                 PSHB
00430P 00A9 36                 PSHA
00431P 00AA 7C 0000   A        INC    CRTFLG   ENTER CRITICAL REGION
00432P 00AD C6 80     A        LDAB   #$80     BLOCKED STATUS WORD

00434           0000  A        IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
00435P 00AF 86 06     A        LDAA   #$06     TASK TIMER NUMBER
00436                          ENDC

00438           0000  A        IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
00439                       LDA A #$0D  TASK TIMER NUMBER
00440                       ENDC

00442P 00B1 BD 0000   A        JSR    XCHSTS   RELACE STATUS WORD ROUTINE
00443P 00B4 7F 17A5   D        CLR    TIME1    CLEAR TIME OUT FLAG
00444P 00B7 7A 0000   A        DEC    CRTFLG   EXIT CRITICAL REGION
00445P 00BA 32                 PULA
00446P 00BB 33                 PULB
00447P 00BC 39                 RTS
```

```
00449                   *  WAIT FOR BUFFER

00451                   *  THIS ROUTINE WILL WAIT FOR A BUFFER TO BECOME INVALID.  IT WILL
00452                   *  CYCLE IN A DYNAMIC LOOP.  THAT IS, IT WILL PASS CONTROL TO THE
00453                   *  SCHEDULER FOR EACH CYCLE THE ROUTINE FINDS THE BUFFER
00454                   *  VALID.  THE INDEX REGISTER RECEIVES THE ADDRESS OF THE PRIMARY
00455                   *  POINTER.  IT IS STACKED FOR FUTURE REFERENCE BY THE WAIT LOOP.
00456                   *  ONCE THE ROUTINE DETECTS AN INVALID BUFFER, THE STACK IS CLEARED
00457                   *  AND CONTROL PASSES BACK TO THE CALLER.  THE ACCUMULATORS
0045B                   *  ARE RESTROED.


00460          00BD  P WAITBF EQU    *
00461P 00BD 36                PSHA
00462P 00BE 37                PSHB
00463P 00BF 0F                SEI             SET FOR CRITICAL REGION
00464P 00C0 7C 0000   A       INC    CRTFLG
00465P 00C3 FF 177B   D       STX    TEMPX    STORE INDEX TO STACK IT
00466P 00C6 B6 1779   D       LDAA   TEMPX+1  FETCH LSB OF ADDRESS
00467P 00C9 F6 177B   D       LDAB   TEMPX    FETCH MSB OF ADDRESS
0046BP 00CC 7A 0000   A       DEC    CRTFLG   CLEAR CRITICAL REGION
00469P 00CF 0E                CLI
0047OP 00D0 36                PSHA            STACK LSB ON STACK
00471P 00D1 37                PSHB            STACK MSB ON STACK

00473                   *  CYCLIC DYNAMIC LOOP.

00475P 00D2 30          WAITAG TSX             FETCH STACK POINTER
00476P 00D3 EE 00     A       LDX    0,X      FETCH PRIMARY ADDRESS POINTER
00477P 00D5 EE 00     A       LDX    0,X      FETCH PRIMARY POINTER
0047BP 00D7 A6 00     A       LDAA   0,X      FETCH BUFFER STATUS WORD
00479P 00D9 27 03 00DE        BEQ    *+5      BUFFER VALID IF VALUE IS ZERO
004BOP 00DB 3F                SWI             RETURN CONTROL TO SCHEDULER
004B1P 00DC 20 F4 00D2        BRA    WAITAG   REPEAT CYCLIC LOOP

004B3                   *  BUFFER IS NOW INVALID.

004B5P 00DE 31                INS             CLEAR STACK
004B6P 00DF 31                INS
004B7P 00E0 33                PULB            RESTORE REGISTERS
004BBP 00E1 32                PULA
004B9P 00E2 39                RTS
```

```
00491                  ************************************************************************
00492                  *                                                                      *
00493                  *    TIMER PROCESS                                                     *
00494                  *                                                                      *
00495                  *    THIS IS THE TIMER TASK.  ITS SOLE PURPOSE IN LIFE IS TO           *
00496                  *    COUNT DOWN A REGISTER ANO SIGNAL WHEN THROUGH.  THE REGISTER      *
00497                  *    IS (CPTIM1) AND IS INITIALIZED FROM (TIMCNT) VIA THE (START1)     *
0049B                  *    ROUTINE.  FOR EACH PASS OF THIS PROCESS, THE COUNTER WILL BE      *
00499                  *    DECREMENTED BY (127 DECIMAL).  THIS ROUTINE WILL UPDATE THE       *
00500                  *    STATUS WORD OF THE PROCESS TO THE MAXIMUM DELAY VALUE.            *
00501                  *    WHEN THE ROUTINE CLOCKS DOWN THE REGISTER, IT WILL SET THE        *
00502                  *    FLAG (TIME1).  THE PROCESS WILL BLOCK ITSELF.                     *
00503                  *                                                                      *
00504                  ************************************************************************



00506          0000  A        IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
00507          00E3  P TKPC6  EQU    *
0050B                         ENDC

00510          0000  A        IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
00511                  TKPCD EQU *
00512                    ENDC


00514P 00E3 0F         TIMER  SEI             SET FOR CRITICAL REGION
00515P 00E4 7C 0000  A        INC    CRTFLG
00516P 00E7 FE 176A  D        LDX    CPTIM1   FETCH THE TIMER REGISTER
00517P 00EA 27 32 011E        BEQ    TIMOUT   IF VALUE IS ZERO TIMEOUT OCCURRED

00519P 00EC F6 176A  D        LDAB   CPTIM1   FETCH MSB OF TIMER REGISTER
00520P 00EF 26 07 00FB        BNE    GREAT    TIMER VALUE GREATER THAN 256 IF VALUE IS NON-ZERO

00522                  *  IF TIMER REGISTER IS LESS THAN 127 THEN HANDLE
00523                  *  SITUTATION UNDER SPECIAL TERMINAL CONDITION.

00525P 00F1 B6 176B  D        LDAA   CPTIM1+1
00526P 00F4 B1 7F    A        CMPA   #$7F
00527P 00F6 23 1D 0115        BLS    LST127   TERMINAL SITUATION IF LESS THAN OR EQUAL TO 127

00529                  *  TIMER VALUE GREATER THAN 127

00531P 00FB B6 176B  D GREAT  LDAA   CPTIM1+1
00532P 00FB F6 176A  D        LDAB   CPTIM1
00533P 00FE B0 7F    A        SUBA   #127     SUBTRACT 127 FROM REGISTER (CPTIM1)
00534P 0100 C2 00    A        SBCB   #0
00535P 0102 B7 176B  D        STAA   CPTIM1+1 UPDATE TIMER REGISTER
00536P 0105 F7 176A  D        STAB   CPTIM1
```

```
00538                   *  READY TO REPLACE STATUS WORD WITH EITHER (127 DECIMAL) OR
00539                   *  A VALUE LESS THAN THAT WHEN ENTERING THROUGH THE LABEL
00540                   *  (SETSTS).  LOAD ACCUMULATOR A WITH THE TASK NUMBER OF
00541                   *  EITHER (06 [DTE]) OR (0D HEX [DCE]).

00543P 0108 C6 7F     A         LDAB   #127      LARGEST DELAY VALUE

00545          0000   A         IFEQ   CHANGE    ASSEMBLE FOR (DTE) IF VALUE IS ZERO
00546P 010A 86 06     A SETSTS  LDAA   #$06      TASK NUMBER FOR (DTE)
00547                           ENDC

00549          0000   A         IFNE   CHANGE    ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
00550                   SETSTS LDA A #$0D  TASK NUMBER FOR (DCE)
00551                     ENDC

00553P 010C BD 0000   A         JSR    XCHSTS    CHANGE STATUS WORD ROUTINE
00554P 010F 7A 0000   A         DEC    CRTFLG    EXIT FROM CRITICAL REGION
00555P 0112 3F                  SWI              RETURN CONTROL TO SCHEDULER
00556P 0113 20 CE 00E3          BRA    TIMER     REPEAT TIMER PROCESS AND CONTINUE TO COUNT DOWN


00558                   *  VALUE OF COUNTER REGISTER IS LESS THAN (127 DECIMAL).
00559                   *  THE REMAINING VALUE IS LOADED INTO THE STATUS WORD FOR
00560                   *  THE DELAY ACTION AND (CPTIM1) IS CLEARED.

00562P 0115 F6 176B   D LST127  LDAB   CPTIM1+1  RETRIEVE LS8 OF COUNTER
00563P 0118 7F 176B   D         CLR    CPTIM1+1  CLEAR COUNTER REGISTER
00564P 011B 5A                  DECB
00565P 011C 20 EC 010A          BRA    SETSTS    SET STATUS WORD FOR THE LAST TIME


00567                   *  THE PROCESS HAS BEEN AWAKEN TO FIND THAT THE COUNTER
00568                   *  REGISTER (CPTIM1) IS ZERO.  A TIMEOUT HAS OCCURRED.
00569                   *  SET THE (TIME1) FLAG INDICATING THE CONDITION.

00571P 011E 7C 17A5   D TIMOUT  INC    TIME1     SET TIMEOUT FLAG
00572P 0121 C6 80     A         LDAB   #$80      LOAD WITH BLOCKED STATUS WORD
00573P 0123 20 E5 010A          BRA    SETSTS    BLOCK THIS PROCESS PERMANENTLY
```

```
00575                  ************************************************************************
00576                  *                                                                      *
00577                  *    PRIORITY TASK                                                     *
00578                  *                                                                      *
00579                  *    THIS IS THE PRIORITY TASK.  THE STATUS WORD IS INTERROGATED       *
00580                  *    BEFORE ANY OF THE OTHER TASKS ARE.  CONSEQUENTLY IT MAY BE        *
00581                  *    EXECUTED FREQUENTLY.  THIS TASK WAS DESIGNED INTO THE SYSTEM      *
00582                  *    AS AN OPTION.  IT IS NOT USED AT THIS TIME.                       *
00583                  *                                                                      *
00584                  ************************************************************************



00586                  *  IF THIS TASK IS ENTERED, CONTROL WILL IMMEDIATELY BE
00587                  *  RETURNED.  HOWEVER, IT WILL NOT 8LOCK ITSELF.

00589           0125  P TKPCM  EQU    *
00590P 0125 3F                 SWI             RETURN CONTROL TO THE SCHEDULER
00591P 0126 20 FD 0125         8RA     TKPCM   REPEAT LOOP IF NECESSARY
```

```
00593                  ************************************************************************
00594                  *                                                                      *
00595                  *    INITIATOR                                                         *
00596                  *                                                                      *
00597                  *    THIS PROCESS ISSUES DATA TO THE TRANSMITTER FOR SENDING TO THE    *
00598                  *    NETWORK.  IT WILL ISSUE AS MAY AS (KCOUNT) PACKETS BEFORE AN      *
00599                  *    ACKNOWLEDGEMENT IS DUE.  THE TABLE QUEUE (INTABL) HOLDS ALL OF    *
00600                  *    THE OUTSTANDING PACKETS AND A FEW OF THE NEW PACKETS.  AS LONG    *
00601                  *    AS ACKNOWLEDGEMENTS COME IN, THE INITIATOR WILL STAY IN THE       *
00602                  *    NORMAL EXECUTION LOOP.  TROUBLE DEVELOPS WHEN THE NORMAL          *
00603                  *    SEQUENCE IS INTERRUPTED FOR SOME REASON.                          *
00604                  *                                                                      *
00605                  *    THE MAIN FLOW OF THIS PROCESS IS TO LOOP INTERROGATING            *
00606                  *    THE INTERPROCESS COMMUNICATION BUFFERS.  THE ORDER OF THE         *
00607                  *    INTERROGATION IS THE (INDATA) BUFFERS, FOLLOWED BY THE (OUTCMD)   *
00608                  *    AND (INRESP) BUFFERS, RESPECTIVELY.  AFTER EACH PASS OF A NORMAL  *
00609                  *    CYCLE COMPLETES, CONTROL WILL PASS TO THE SCHEDULER.              *
00610                  *                                                                      *
00611                  ************************************************************************



00613          0000  A        IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
00614          012B  P TKPC0   EQU    *
00615                         ENDC

00617          0000  A        IFNE   CHANGE   ASSMBLE FOR (DCE) IF VALUE IS NON-ZERO
00618                  TKPC7 EQU *
00619                    ENDC


00621                  *  INTERROGATE (INDATA) BUFFERS

00623                  *  FIRST, LOOK AT THE SECONDARY BUFFER.  PASS BY THIS SECTION
00624                  *  IF IT IS VALID.  OTHERWISE, LOOK AT THE PRIMARY BUFFER.
00625                  *  IF IT SHOWS A VALID STATE, SWAP THE PRIMARY AND SECONDARY
00626                  *  POINTER.

0062BP 012B FE 1754  D INORML LDX    STDBY1   FETCH THE SECONDARY POINTER
00629P 012B A6 00    A        LDAA   0,X      TEST FOR VALIDITY
00630P 012D 26 1E 014D        BNE    ISKIP1   VALID BUFFER IF VALUE IS NON-ZERO
```

```
00632                   *  THE SECONDARY BUFFER IS INVALID.  CHECK THE PRIMARY BUFFER.

00634P 012F FE 1752  D          LDX    INDATA   FETCH THE PRIMARY POINTER
00635P 0132 A6 00    A          LDAA   0,X      TEST FOR VALIDITY
00636P 0134 27 27 0150          BEQ    ISKIP2   VALID BUFFER IF VALUE IS NON-ZERO

0063B                   *  SECONDARY BUFFER EMPTY, PRIMARY BUFFER FULL.
00639                   *  SWAP THE PRIMARY AND SECONDARY BUFFERS.

00641P 0136 0F                  SEI             SET FOR CRITICAL REGION
00642P 0137 7C 0000  A          INC    CRTFLG
00643P 013A FF 176E  D          STX    TEMPR2
00644P 013D FE 1754  D          LDX    STDBY1
00645P 0140 FF 1752  D          STX    INDATA
00646P 0143 FE 176E  D          LDX    TEMPR2
00647P 0146 FF 1754  D          STX    STDBY1
0064BP 0149 7A 0000  A          DEC    CRTFLG   EXIT CRITICAL REGION
00649P 014C 0E                  CLI



00651                   *  INPUT A PACKET

00653                   *  THERE IS A PACKET WAITING TO BE QUEUED IN THE (INTABL) TABLE.
00654                   *  CHECK TO SEE IF THERE IS ROOM IN THE QUEUE.  IF THERE IS, QUEUE
00655                   *  IT.  DETERMINING THE AVAILABILITY OF SPACE IN THE TABLE IS
00656                   *  ACHIEVED BY COMPARING THE REAR POINTER (NEXT) PLUS ONE
00657                   *  TO THE FRONT POINTER (NPROFR).

00659P 0140 B6 1791  D ISKIP1 LDAA   NEXT     FETCH REAR POINTER
0065DP 0150 C6 01    A        LDAB   #1
00661P 0152 BD 0101  P        JSR    MODULO   ADD ONE TO THE REAR POINTER MODULO 8
00662P 0155 B1 17BE  D        CMPA   NPROFR   COMPARE REAR POINTER PLUS ONE WITH FRONT POINTER
00663P 015B 27 03 0150        BEQ    ISKIP2   QUEUE FULL IF VALUE IS ZERO
00664P 015A BD 0109  P        JSR    ACPTIN   QUEUE THE DATA PACKET
```

```
00666                   *  TRASNMIT A DATA PACKET IF POSSIBLE

00668                   *  A DATA PACKET IS TRANSMITTED IF THE QUEUE HAS ONE AVAILABLE,
00669                   *  THE (DCE) NODE IS NOT BUSY, AND THE (OUTCMD) PRIMARY BUFFER
00670                   *  IS EMPTY OR INVALID.



00672P 015D 86 1788  D ISKIP2 LDAA   NBUSY    FETCH STATUS OF (DCE) INTERFACE
00673P 0160 26 28 018A        BNE    ISKP3A   INTERFACE IS BUSY IF VALUE IS NON-ZERO


00675                   *  TEST FOR QUEUE HAVING A PACKET IN IT.  IF IT DOES, CHECK
00676                   *  TO SEE IF THE OUTSTANDING PACKET LIMITATIONS HAVE
00677                   *  BEEN EXCEEDED.

00679P 0162 B6 178C  D        LDAA   VOFS     FETCH FRONT POINTER FOR TRANSMISSION PURPOSES
00680P 0165 B1 1791  D        CMPA   NEXT     COMPARE WITH REAR POINTER
00681P 0168 27 20 018A        BEQ    ISKP3A   PACKET TO TRANSMIT IF VALUE IS NON-ZERO

00683                   *  TEST FOR LIMITATION

00685P 016A B0 178E  D        SUBA   NPROFR   SUBTRACT REAR POINTER
00686P 016D 2A 02 0171        BPL    *+4      TEST FOR COMPENSATION MEASURE
00687P 016F BB 08    A        ADDA   #8       COMPENSATE IF FRONT ½ REAR
00688P 0171 B1 07    A        CMPA   #KCONST  CONPARE WITH LIMITING FACTOR
00689P 0173 2C 15 018A        BGE    ISKP3A   LIMIT EXCEEDED IF BRANCH IS TAKEN


00691                   *  CHECK TO SEE IF THE PRIMARY BUFFER (OUTCMD) IS
00692                   *  INVALID.  IF IT IS, TRANSMIT PACKET.

00694P 0175 FE 1756  D        LDX    OUTCMD   FETCH PRIMARY BUFFER POINTER
00695P 0178 A6 00    A        LDAA   0,X      TEST VALIDITY
00696P 017A 26 0E 018A        BNE    ISKP3A   BUFFER IS EMPTY IF VALUE IS ZERO

00698                   *  TRANSMIT A DATA PACKET AND ADVANCE THE POINTER (VOFS).

00700P 017C BD 01FD  P        JSR    XMITPK   DEQUEUE THE DATA AND TRANSMIT
00701P 017F F6 178C  D        LDAB   VOFS     FETCH POINTER TO INCREMENT
00702P 0182 B6 01    A        LDAA   #1
00703P 0184 BD 01D1  P        JSR    MODULO   ADVANCE POINTER (VOFS) BY ONE
00704P 0187 B7 178C  D        STAA   VOFS     UPDATE POINTER
```

```
00706                   *  INTERROGATE (INRESP) BUFFER

00708                   *  LOOK FOR AN INCOMING RESPONSE OF SOME SORT.
00709                   *  IF ONE EXISTS, EXCHANGE THE PRINARY ANO SECONDARY
00710                   *  POINTERS.  ELSE, SKIP THIS SECTION.



00712P 01BA FE 175A  0 ISKP3A LDX    INRESP   FETCH PRIMARY POINTER FOR (INRESP)
00713P 01BD A6 00    A        LDAA   0,X      TEST FOR VALIDITY
00714P 01BF 27 1A 01AB        BEQ    ISKIP4   A FULL BUFFER EXISTS IF THE VALUE IS NON-ZERO

00716                   *  EXCHANGE THE PRIMARY AND SECONDARY BUFFERS

00718P 0191 0F                SEI             SET FOR CRITICAL REGION
00719P 0192 7C 0000  A        INC    CRTFLG
00720P 0195 FF 176E  D        STX    TEMPR2
00721P 0198 FE 175C  D        LDX    STDBY3
00722P 019B FF 175A  D        STX    INRESP
00723P 019E FE 176E  D        LDX    TEMPR2
00724P 01A1 FF 175C  D        STX    STDBY3
00725P 01A4 7A 0000  A        DEC    CRTFLG   EXIT CRITICAL REGION
00726P 01A7 0E                CLI
00727P 01AB BD 0245  P        JSR    RECRPC   DECODE RESPONCE AND ACT ACCORDINGLY


00729                   *  CHECK TIMEOUT

00731                   *  A TIMEOUT HAS OCCURRED IF THE FLAG (TIME1) IS
00732                   *  SET TO A NON-ZERO VALUE.  ENTER THE ROUTINE TO
00733                   *  TAKE THE NECESSARY RECOVERY ACTION.

00735P 01AB B6 17A5  D ISKIP4 LDAA   TIME1    FETCH TIMEOUT FLAG
00736P 01AE 27 03 01B3        BEQ    *+5      TIMEOUT HAS OCCURRED IF VALUE IS NON-ZERO
00737P 01B0 BD 042A  P        JSR    T1EXPR   ENTER TIMEOUT RECOVERY ROUTINE

00739P 01B3 3F                SWI             RETURN CONTROL TO SCHEDULER
00740P 01B4 7E 0128  P        JMP    INORML   REPEAT THE PROCESS WHEN CONTROL RETURNS
```

```
00742                    *  FIND POINTER (INTABL)

00744                    *  GIVEN A ONE BYTE POINTER FOR (INTABL), THIS ROUTINE WILL RETURN
00745                    *  THE PHYSICAL POINTER POINTING INTO THE TABLE.  ACCUMULATOR
00746                    *  B IS SAVED.  ACCUMULATOR A RECEIVES THE BYTE POINTER.
00747                    *  THE VARIABLE (TEMPR2) IS USED.  POINTER IS RETURNED IN THE
0074B                    *  INDEX REGISTER.



00750           01B7  P FINDTB EQU    *
00751P 01B7 37                 PSHB           SAVE ACCUMULATOR B
00752P 01B8 CE 1722  D AT      LDX    #INTABL FETCH THE BASE ADDRESS
00753P 01BB FF 176E  D         STX    TEMPR2  STORE IN WORKING REGISTER
00754P 01BE 16                 TAB            MULTIPLY ACC A BY THREE
00755P 01BF 4B                 ASLA
00756P 01C0 1B                 ABA
00757P 01C1 BB 176F  D         ADDA   TEMPR2+1 ADD DISPLACEMENT TO LSB OF BASE ADDRESS
0075BP 01C4 B7 176F  D         STAA   TEMPR2+1 UPDATE WORKING REGISTER
00759P 01C7 24 03 01CC         BCC    *+5     COMPENSATE FOR CARRY
00760P 01C9 7C 176E  D         INC    TEMPR2  INCREMENT MSB IF CARRY IS PRESENT
00761P 01CC FE 176E  D         LDX    TEMPR2  FETCH PHYSICAL ADDRESS FROM WORKING REGISTER
00762P 01CF 33                 PULB           RESTORE ACCUMULATOR B
00763P 01D0 39                 RTS



00765                    *  ADD EIGHT (MODULO B)

00767                    *  THIS IS THE ADDITION ROUTINE USED BY THE INITIATOR.
0076B                    *  IT WILL ADD THE TWO NUMBERS IN ACCUMULATORS A AND B.
00769                    *  IT USES MODULO B ADDITION.  THE RESULT IS CONTAINED
00770                    *  IN ACCUMULOATOR A.



00772           01D1  P MODULO EQU    *
00773P 01D1 1B                 ABA            CONVENTIONAL ADD
00774P 01D2 B1 07    A         CMPA   #7      CHECK FOR OVERFLOW
00775P 01D4 2F 02 01DB         BLE    *+4     OVERFLOW DOES NOT EXIST IF BRANCH IS TAKEN
00776P 01D6 B0 0B    A         SUBA   #B      COMPENSATE FOR MODULO EIGHT
00777P 01DB 39                 RTS
```

```
00779                    *  QUEUE DATA (INTA8L)

00781                    *  THE INFORMATION THAT IS CONTAINED IN THE SECONDARY 8UFFER
00782                    *  OF (INDATA) IS QUEUED INTO THE (INTA8L) TA8LE.  THE NODE
00783                    *  ADDRESS AND THE 8YTE COUNT ARE QUEUED.  THE ACTUAL DATA IS
00784                    *  NOT.  THE SECONDARY 8UFFER ADDRESS OF (INDATA) IS PASSED AS
00785                    *  AN ARGUMENT IN THE INDEX REGISTER.



00787           01D9  P ACPTIN EQU    *
00788P 01D9 A6 03     A        LDAA   3,X      FETCH THE LS8 OF THE NODE ADDRESS
00789P 01D8 E6 02     A        LDA8   2,X      FETCH THE MS8 OF THE NODE ADDRESS
00790P 01DD 37                 PSH8
00791P 01DE 36                 PSHA
00792P 01DF E6 01     A        LDA8   1,X      FETCH THE 8YTE COUNT
00793P 01E1 6F 00     A        CLR    0,X      DEVALIDATE THE SECONDARY 8UFFER OF (INDATA)

00795                    *  FIND TA8LE POSITION.

00797P 01E3 86 1791   D        LDAA   NEXT     FETCH REAR POINTER
00798P 01E6 8D 0187   P        JSR    FINDT8   CALCULATE THE PHYSICAL ADDRESS
00799P 01E9 E7 00     A        STA8   0,X      ENTER INTO THE TA8LE THE 8YTE COUNT
00800P 01E8 32                 PULA
00801P 01EC 33                 PUL8
00802P 01ED E7 01     A        STA8   1,X      ENTER INTO THE TA8LE THE LS8 OF THE NODE ADDRESS
00803P 01EF A7 02     A        STAA   2,X      ENTER INTO THE TA8LE THE MS8 OF THE NODE ADDRESS

00805                    *  ADVANCE REAR POINTER 8Y ONE

00807P 01F1 86 1791   D        LDAA   NEXT     FETCH REAR POINTER
00808P 01F4 C6 01     A        LDA8   #1
00809P 01F6 8D 01D1   P        JSR    MODULO   ADD USING MODULO 8 FORMAT
00810P 01F9 87 1791   D        STAA   NEXT     UPDATE REAR POINTER
00811P 01FC 39                 RTS
```

```
00813                    *  DEQUEUE OATA (INTA8L)

00815                    *  THIS ROUTINE WILL DEQUEUE THE DATA PACKET FROM (INTA8L)
00816                    *  AND INPUT THE NECESSARY INFORMATION INTO THE PRIMARY
00817                    *  8UFFER OF (OUTCMO).  THE PACKET THAT IS 8EING TRANSMITTED
00818                    *  IS AN INFORMATION PACKET.  WHEN THIS COMPLETES, THE
00819                    *  TIMER IS STARTEO.



00821           01FD  P XMITPK EQU    *
00822P 01FD B6 178C  D         LDAA   VOFS     FETCH THE FRONT POINTER OF (INTA8L)
00823P 0200 8D 0187  P         JSR    FINDT8   CALCULATE THE PHYSICAL ADDRESS POINTER
00824P 0203 A6 00    A         LDAA   0,X      FETCH THE 8YTE COUNT FROM THE TA8LE
00825P 0205 36                 PSHA
00826P 0206 A6 02    A         LOAA   2,X      FETCH THE LS8 OF THE NODE POINTER
00827P 0208 E6 01    A         LDA8   1,X      FETCH THE MS8 OF THE NODE POINTER

00829P 020A FE 1756  D         LDX    OUTCMD   FETCH THE PRIMARY 8UFFER POINTER
00830P 020D A7 07    A         STAA   7,X      INITIALIZE 8UFFER WITH THE LS8 OF THE NODE POINTER
00831P 020F E7 06    A         STA8   6,X      INITIALIZE 8UFFER WITH THE MS8 OF THE NODE POINTER
00832P 0211 32                 PULA
00833P 0212 A7 02    A         STAA   2,X      INITIALIZE THE 8UFFER WITH THE 8YTE COUNT
00834P 0214 86 179A  D         LDAA   POLL8T   FETCH THE POLL 8IT INOICATOR OF THE INITIATOR
00835P 0217 A7 03    A         STAA   3,X      INITIALIZE THE 8UFFER WITH THE POLL 8IT
00836P 0219 F6 178C  D         LDA8   VOFS     FETCH THE SEQUENCE NUM8ER OF THE OATA PACKET
00837P 021C E7 04    A         STA8   4,X      INITIALIZE THE 8UFFER WITH THE SEQUENCE NUM8ER
```

```
00839                   *  TEST THE STATE OF (NREJ).  THIS WILL OETERMINE
00840                   *  THE STATE OF THE INITIATOR.  IF SET, THE INITIATOR
00841                   *  IS TRYING TO RECOVER FROM A REJECTION STATE.
00842                   *  COMPARE THE SEQUENCE NUM8ER 8EING TRANSMITTEO WITH THE
00843                   *  REJECTED VALUE.  IF THEY MATCH, THE INITIATOR HAS RECOVEO.



00845P 021E B6 1789  D        LDAA   NREJ     FETCH NODE REJECTION FLAG
00846P 0221 27 08 022B        BEQ    SKPK1    NORMAL CONOITION IF VALUE IS ZERO
00847P 0223 F1 1787  D        CMPB   REJVAL   COMPARE WITH REJECTEO VALUE
00848P 0226 26 03 022B        BNE    *+5      IF VALUE IS ZERO, RECOVEREO
00849P 0228 7F 1789  D        CLR    NREJ     CLEAR FOR NORMAL STATE


00851                   *  TEST FOR A TIME RECOVERY STATE.  IF IT EXISTS, INITIATOR HAS
00852                   *  TIMEO OUT ANO RETRANSMISSION IS UNOER WAY.  THE TIME OUT STATE
00853                   *  IS CLEARED IF THE SEQUENCE NUM8ER 8EING TRANSMITTEO IS EQUAL
00854                   *  TO THE LAST ACKNOWLEDGEO SEQUENCE NUM8ER.


00856P 022B B6 1794  D SKPK1  LOAA   TBUSY    FETCH TIME OUT RECOVERY STATE FLAG
00857P 022E 27 0B 023B        BEQ    SKPK2    INITIATOR IS IN NORMAL STATE IF VALUE IS ZERO
00858P 0230 F6 17A6  0        LOAB   XVAR     FETCH LAST ACKNOWLEOGED SEQUENCE NUMBER
00859P 0233 F1 178C  D        CMPB   VOFS     COMPARE WITH CURREN SEQUENCE NUMBER
00860P 0236 26 03 023B        BNE    *+5      INITIATOR RETURNS TO NORMAL STATE IF VALUE IS ZERO
00861P 0238 7F 1794  D        CLR    TBUSY    RETURN TO NORMAL STATE


00863                   *  CONTROL ENOS UP HERE.  THE INITIALIZATION OF THE PRIMARY
00864                   *  BUFFER IS COMPLETEO HERE.  THE PACKET TYPE IS DECLAREO AS
00865                   *  AN INFORMATION PACKET ANO THE BUFFER IS VALIOATEO.


00867P 023B 86 01    A SKPK2  LDAA   #1       LOAO WITH PACKET TYPE
00868P 023D A7 01    A        STAA   1,X      INITIALIZEO BUFFER WITH PACKET TYPE
00869P 023F A7 00    A        STAA   0,X      VALIOATE BUFFER WITH NON-ZERO VALUE
00870P 0241 B0 008E  P        JSR    START1   START TIMER FOR THIS PACKET
00871P 0244 39                RTS
```

```
00873                   *  DECODE RECEIVED PACKET

00875                   *  A PACKET HAS BEEN RECEIVED AND THIS ROUTINE WAS CALLED.  THIS
00876                   *  ROUTINE WILL DECODE THE PACKET TYPE AND PASS CONTROL TO THE
00877                   *  APPROPRIATE ROUTINE.  THE PACKET THAT IS TO BE RECEIVED IS A
00878                   *  RESPONSE.  IF A COMMAND IS RECEIVED, AN ERROR ROUTINE WILL
00879                   *  BE ENTERED.  IF THE INCOMING FINAL BIT DOES NOT MATCH APPROPRIATELY
00880                   *  WITH THE INITATOR POLL BIT, AN ERROR WILL OCCUR.  OTHERWISE,
00881                   *  ITS A RESPONSE AND CAN BE DECODED AS SUCH.



00883          0245  P RECRPC EQU    *
00884P 0245 A6 01    A         LDAA   1,X      FETCH PACKET TYPE FROM SECONDARY BUFFER
00885P 0247 2B 20 0269         BMI    IRSREJ   IF NEGATIVE, A RECEPTION ERROR HAS OCCURRED
00886P 0249 E6 03    A         LDAB   3,X      FETCH THE POLL BIT
00887P 024B 27 05 0252         BEQ    ISKIP5   CONDITION IS NORMAL IF VALUE IS ZERO

00889                   *  CONDTION IS SPECIAL, TEST IT AGAINST THE ALREADY
00890                   *  EXISTING POLL BIT OF THE INITIATOR.

00892P 024D F6 179A  D         LDAB   POLLBT   FETCH POLL BIT OF THE INITIATOR
00893P 0250 27 20 0272         BEQ    INVLRS   NORMAL CONDITION IF POLL BIT OF INITIATOR IS ZERO

00895                   *  VALID RESPONSES ARE 4 THROUGH 8.

00897P 0252 81 03    A ISKIP5  CMPA   #3       COMPARE FOR COMMAND
00898P 0254 2F 1C 0272         BLE    INVLRS   A COMMAND HAS A VALUE OF LESS THAN 4
00899P 0256 81 08    A         CMPA   #8       COMPARE FOR A CMDR RESPONSE OR GREATER VALUE
00900P 0258 2C 18 0272         BGE    INVLRS   ERROR IF A CMDR OR GREATER IS RECEIVED

00902                   *  VALID RESPONSE - DECODE IT.

00904P 025A 81 04    A         CMPA   #4       TEST FOR A RR RESPONSES
00905P 025C 27 0E 026C         BEQ    IRSRR    TEST POSITIVE IF VALUE IS ZERO

00907P 025E 81 05    A         CMPA   #5       TEST FOR A RNR RESPONSE
00908P 0260 27 0D 026F         BEQ    IRSRNR   TEST IS POSITIVE IF VALUE IS ZERO

00910P 0262 81 06    A         CMPA   #6       TEST FOR REJ RESPONSE
00911P 0264 27 03 0269         BEQ    IRSREJ   TEST IS POSITIVE IF VALUE IS ZERO

00913                   *  DEFAULT VALUE IS 7.  U.A. RESPONSE.

00915P 0266 6F 00    A         CLR    0,X      DEVALIDATE BUFFER FOR RESPONSE OF 7
00916P 0268 39                 RTS


00918                   *  EXTENSION TABLE

00920P 0269 7E 037C  P IRSREJ  JMP    RESREJ   EXTENSION TO REJECT RESPONSE ROUTINE
00921P 026C 7E 0332  P IRSRR   JMP    RESPRR   EXTENSION TO READY RESPONSE ROUTINE
00922P 026F 7E 03A7  P IRSRNR  JMP    RESRNR   EXTENSION TO NOT READY RESPONSE ROUTINE
```

```
00924                   *  INVALIO RESPONSE

00926                   *  IF AN INVALIO RESPONSES OCCURS, ALL HELL 8REAKS LOOSE.
00927                   *  THE INITIATOR ASSUMES THAT SOMETHING SERIOUS HAS GONE WRONG,
00928                   *  A NON-RECOVERA8LE ERROR.  THE INITIATOR CALLS ON THIS ROUTINE
00929                   *  TO ISSUE THE SARM COMMANO.  THE SARM COMMANO INITIALIZES
00930                   *  8OTH THE INITIATOR ANO RESPONOER ON THE OTHER SIOE.  IF
00931                   *  A U.A. RESPONSE IS NOT RECEIVEO, A OISCONNECT COMMANO WILL
00932                   *  8E ISSUEO.  AT THIS POINT, THE CHANNEL IS OOWN.  HOWEVER,
00933                   *  A REPEAT ACTION OCCURS ANO IT IS ASSUMEO THE LINK IS
00934                   *  AUTOMATICALLY RECOVEREO.  THIS IS A FUNCTION OF THIS SYSTEM
00935                   *  ANO IS NOT PART OF X.25.

00937          0272  P INVLRS EQU    *
00938P 0272 C6 02    A         LOA8   #2        LOAO WITH COOEO COMMANO WORO FOR AN SARM
00939P 0274 80 0286  P         JSR    SAROSC    ISSUE SARM COMMANO
00940P 0277 25 2F 02A8         8CS    TRYOSC    CARRY 8IT SET IF NO RESPONSE TO SARM COMMANO

00942                   *  REPSONSE IS MAOE
00943                   *  RETURN ALL OUTSTANOING NOOES TO THE NOOE POOL.

00945P 0279 F6 178E  O ILOOP   LOA8   NPROFR    FETCH LAST ACKNOWLEGEO PACKET
00946P 027C F1 1791  O         CMP8   NEXT      COMPARE IT WITH THE FRONT POINTER
00947P 027F 27 13 0294         8EQ    IRETRY    QUEUE IS EMPTY IF VALUE IS ZERO
00948P 0281 17                 T8A
00949P 0282 80 0187  P         JSR    FINOT8    FINO PHYSICAL AOORESS OF OUTSTANONG PACKET
00950P 0285 EE 01    A         LOX    1,X       FETCH THE NOOE AOORESS FROM (INTA8L)
00951P 0287 80 0076  P         JSR    PUTNOO    RETURN NOOE TO POOL

00953                   *  INCREASE (VOFS) 8Y ONE

00955P 028A 86 01    A         LOAA   #1
00956P 028C 80 0101  P         JSR    MOOULO    AOO ONE TO (VOFS) MOOULO 8 FORMAT
00957P 028F 87 178E  O         STAA   NPROFR    UPOATE ACKNOWLEOGEMENT POINTER
00958P 0292 20 E5 0279         8RA    ILOOP     REPEAT LOOP


00960                   *  RESET ALL POINTERS WHICH WILL FLUSH OUT THE
00961                   *  8UFFERS ANO RETURN TO A NORMAL SITUATION.

00963P 0294 4F          IRETRY CLRA              CLEAR THE FOLLOWING POINTERS:
00964P 0295 87 178C  O         STAA   VOFS      FRONT (INTA8L) POINTER
00965P 0298 87 178E  O         STAA   NPROFR    LAST ACKNOWLEOGEMENT POINTER
00966P 0298 87 1793  O         STAA   RCOUNT    REPEAT COUNT
00967P 029E 87 179A  O         STAA   POLL8T    INITIATOR POLL 8IT
00968P 02A1 87 1791  O         STAA   NEXT      REAR POINTER FOR (INTA8L)
00969P 02A4 87 17A5  O         STAA   TIME1     TIMER FLAG
00970P 02A7 39                 RTS
```

```
00972                   *  ISSUE DISCONNECT

00974                   *  THIS ROUTINE IS CALLED FROM (INVLRS) WHEN AN ACKNOWLEDGEMENT HAS
00975                   *  NOT BEEN RECEIVED FOR AN SARM COMMAND.  THIS ROUTINE WILL ISSUE
00976                   *  A DISCONNECT COMMAND FORCING THE LINK BETWEEN THE INITIATOR AND
00977                   *  RESPONDER TO GO DOWN.  AT THIS POINT, A NON-RECOVERABLE SITUATION
00978                   *  EXISTS.  HOWVER, CONTROL WILL CYCLE AND FLUSH OUT THE BUFFERS
00979                   *  AS IF THE LINK WAS COMING UP.  THIS ACTION IS NOT A FUNCTION OF
00980                   *  RECOMMENDATION X.25.

00982           02A8  P TRYDSC EQU    *
00983P 02A8 86 03     A        LDAA   #3        LOAD WITH CODED VALUE FOR DISCONNECT COMMAND
00984P 02AA BD 0286   P        JSR    SARDSC    CALL ROUTINE TO ISSUE COMMAND
00985P 02AD BD 0279   P        JSR    ILOOP     RETURN ALL NODES TO THE NODE POOL
00986P 0280 7F 17A1   D        CLR    HONLNX    INDICATE THAT THE LINK HAS GONE DOWN

00988P 0283 3F                 SWI              RETURN CONTROL TO THE SCHEDULER
00989P 0284 20 DE 0294         BRA    IRETRY    RETURN TO FLUSH OUT BUFFER AND RETART LINK
```

```
00991                   *  ISSUE SARM/DISCONNECT COMMANDS

00993                   *  THIS ROUTINE CONSTITUTES THE ABNORMAL LOOP.  CONTROL PASSES TO
00994                   *  THIS POINT WHEN A CRITICAL SITUATION EXISTS, I.E., NO DATA BEING
00995                   *  RECEIVED.  THIS ROUTINE WILL ISSUE THE APPROPRIATE COMMAND ANO
00996                   *  CYCLE WAITING  TO RECEIVE A RESPONSE.  IF A TIME OUT EXPIRES,
00997                   *  RETRANSMISSION WILL OCCUR WITH THE POLL BIT SET.  THE REPEATING
0099B                   *  WILL CONTINUE FOR (N2NUM) TIMES.

01000                   *  CLEAR POLL BIT AND REPEAT COUNT REGARDLESS OF THE PAST.

01002           02B6  P SARDSC EQU    *
01003P 02B6 7F 179A  D         CLR    POLLBT
01004P 02B9 7F 1793  D RESRDC  CLR    RCOUNT
01005P 02BC BD 031F  P WAITIR  JSR    ISSMDS   ISSUE COMMAND TO THE PRINMARY BUFFER OF (OUTCMD)
01006P 02BF FE 175A  D WAITIQ  LDX    INRESP   FETCH PRIMARY BUFFER POINTER
01007P 02C2 A6 00    A         LDAA   0,X      FETCH STATUS WORO
0100BP 02C4 26 1F 02E5         BNE    XXYY     BUFFER VALID IF NON-ZERO
01009P 02C6 B6 17A5  D         LDAA   TIME1    CEHCK FOR TIME OUT POSSIBILITY
01010P 02C9 27 F4 02BF         BEQ    WAITIQ   TIME OUT HAS OCCURRED IF VALUE IS NON-ZERO

01012P 02CB 7F 17A5  D         CLR    TIME1    CLEAR FLAG FOR FUTURE USE
01013P 02CE 7C 1793  D         INC    RCOUNT   INCREMENT REPEAT COUNT
01014P 02D1 B6 1793  D         LDAA   RCOUNT   FETCH REPEAT COUNT
01015P 02D4 B1 04    A         CMPA   #N2NUM   COMPARE TO UPPER REPEAT LIMIT
01016P 02D6 26 E4 02BC         BNE    WAITIR   LIMIT REACHED IF VALUE IS ZERO

01018                   *  UPPER LIMIT REACHED.  SET POLL BIT FOR LAST CHANCE.

01020P 02DB B6 179A  D         LDAA   POLLBT   FETCH POLL BIT
01021P 02DB 26 06 02E3         BNE    *+B      SKIP TO END IF ALREADY SET
01022P 02DD 4C                 INCA            SET POLL BIT TO ONE
01023P 02DE B7 179A  D         STAA   POLLBT   UPDATE POLL BIT
01024P 02E1 20 D6 02B9         BRA    RESRDC   REPEAT RETRANSMISSION WITH POLL BIT SET
01025P 02E3 0D                 SEC             SET CARRY FLAG IF POLL BIT IS SET
01026P 02E4 39                 RTS
```

```
0102B                   *  RECEIVE RESPONSE

01030                   *  THIS ROUTINE IS ENTEREO IF THERE IS A RESPONSE FROM THE
01031                   *  ABNORMAL LOOP.  THE BUFFER POINTERS ARE EXCHANGEO ANO
01032                   *  THE STATUS WORO IS FETCH.  THIS ROUTINE IS LOOKING FOR
01033                   *  ONLY AN U.A. RESPONSE ANO WILL REJECT ALL OTHERS.  IN
01034                   *  AOOITION, THE POLL BITS MUST BE THE SAME OR THE
01035                   *  RESPONSE WILL BE REJECTEO.

01037                   *  EXCHANGE THE PRIMARY ANO SECONOARY POINTERS

01039P 02E5 OF          XXYY   SEI             SET FOR CRITICAL REGION
01040P 02E6 7C 0000  A         INC    CRTFLG
01041P 02E9 FF 176E  O         STX    TEMPR2
01042P 02EC FE 175A  O         LOX    INRESP
01043P 02EF FF 175C  O         STX    STOBY3
01044P 02F2 FE 176E  O         LOX    TEMPR2
01045P 02F5 FF 175A  O         STX    INRESP
01046P 02FB 7A 0000  A         OEC    CRTFLG  EXIT CRITICAL REGION
01047P 02FB OE                 CLI

01049P 02FC A6 01    A         LOAA   1,X     FETCH THE STATUS WORO
0105OP 02FE 2B 1A 031A         BMI    REJRES  REJECT RESPONSE IF THE STATUS WORO IS NEGATIVE
01051P 0300 B1 07    A         CMPA   #7      COMPARE COOE WITH A SEVEN
01052P 0302 26 16 031A         BNE    REJRES  VALIO RESPONSE IF STATUS IS SEVEN

01054P 0304 B6 179A  O         LOAA   POLLBT  FETCH INITIATOR POLL BIT
01055P 0307 27 06 030F         BEQ    ABSKIP  SKIP IF ZERO VALUE
01056P 0309 60 03    A         TST    3,X     COMPARE WITH A NEGATIVE POLL BIT
01057P 030B 27 00 031A         BEQ    REJRES  REJECT RESPONSE IF RECEIVEO POLL BIT IS POSITIVE
0105BP 0300 20 04 0313         BRA    OKSKP   SKIP POSITIVE TEST

01060P 030F 60 03    A ABSKIP  TST    3,X     TEST RECEIVEO POLL BIT AGAINST POSITVE POLL BIT
01061P 0311 26 07 031A         BNE    REJRES  REJECT RESPONSE IF INCOMING POLL BIT IS NEGATIVE
01062P 0313 6F 00    A OKSKP   CLR    O,X     OEVALIOATE THE BUFFER
01063P 0315 BO 00AB  P         JSR    RESET1  VALIO RESPONSE, RESET TIMER
01064P 031B OC                 CLC            CLEAR CARRY FOR POSITIVE CONFIRMATION
01065P 0319 39                 RTS
01066P 031A 6F 00    A REJRES  CLR    O,X     OEVALIOATE BUFFER
01067P 031C 7E 02BC  P         JMP    WAITIR  RETURN TO WAIT LOOP
```

```
01069                    *  ISSUE COMMAND

01071                    *  THIS ROUTINE IS CALL BY THE ABNORMAL LOOP TO ISSUE A
01072                    *  COMMAND TO THE TRANSMITTER.  THE COMMAND WORD IS PASSED
01073                    *  TO THIS ROUTINE VIA ACCUMULATOR B.

01075           031F  P ISSMDS EQU    *
01076P 031F CE 1756  D         LDX    #OUTCMD  FETCH THE PRIMARY POINTER
01077P 0322 BD 00BD  P         JSR    WAITBF   WAIT FOR THE BUFFER TO BE EMPTY

01079                    *  INITIALIZE THE PRIMARY BUFFER FIELDS.

01081P 0325 E7 01    A         STAB   1,X      INITIALIZE BUFFER WITH PACKET TYPE
01082P 0327 B6 179A  D         LDAA   POLLBT   FETCH INITIATOR POLL BIT
01083P 032A A7 03    A         STAA   3,X      INITIALIZE BUFFER WITH POLL BIT
01084P 032C E7 00    A         STAB   0,X      VALIDATE BUFFER

01086P 032E BD 00BE  P         JSR    START1   START TIMER
01087P 0331 39                 RTS
```

```
01089                   *  RECEIVE READY RESPONSE

01091                   *  THIS ROUTINE IS ACCESSED WHEN IT IS DETERMINED THAT A READY
01092                   *  RESPONSE HAS BEEN RECEIVED.  THE ROUTINE WILL ANALYZE
01093                   *  THE RESPONSE TO DETERMINE THE VALIDITY OF IT.



01095          0332  P RESPRR EQU    *
01096P 0332 F6 179A  0         LOAB   POLLBT   FETCH THE POLL BIT
01097P 0335 27 10 0347         BEQ    RRNORM   NORMAL SITUATION IF VALUE IS ZERO


01099                   *  A SPECIAL SITUATION.  THE POLL BIT IS SET.  CHECK TO
01100                   *  SEE IF IT CAN BE CLEARED.  IT WILL BE CLEARED IF THE
01101                   *  RETURNING ACKNOWLEDGEMENT IS EQUAL TO THE LAST
01102                   *  ACKNOWLEDGEMENT BFORE THE SEQUENCE GOT OUT OF SYNC.

01104P 0337 E6 05    A         LOAB   5,X      FETCH ACKNOWLEDGEMENT
01105P 0339 F1 17A6  0         CMPB   XVAR     COMPARE WTIH LAST ACKNOWLEDGEMENT
01106P 033C 26 09 0347         BNE    RRNORM   NORMAL SITUATIDN RE-EXISTS IF VALUE IS ZERO
01107P 033E 5F                 CLRB            CLEAR THE FOLLOWING VARIABLES:
01108P 033F F7 1794  0         STAB   TBUSY    TIME OUT BUSY FLAG
01109P 0342 F7 179A  0         STAB   POLLBT   INITIATOR POLL BIT
01110P 0345 20 04 034B         BRA    RRNRM1   RESUME NORMAL CONDITION
01111P 0347 5F           RRNORM CLRB            CLEAR THE FOLLOWING FLAGS:
01112P 0348 F7 1793  0         STAB   RCOUNT   REPEAT COUNT
01113P 034B F7 1789  0 RRNRM1 STAB   NREJ     REJECT RESPONSE FLAG


01115                   *  CHECK FOR A BUSY NODE.  IF TEST IS POSITIVE, SKIP
01116                   *  ALL THE FAN-FAIR AND UPDATE (VOFS).

01118P 034E F6 1788  0         LOAB   NBUSY    FETCH NODE BUSY FLAG
01119P 0351 26 1B 036E         BNE    RSTVFR   NODE IS BUSY IF SET TO A NON-ZERO VALUE
```

```
01121                   *  ENTRY POINT FOR RECEIVE NOT READY RESPONSE.
01122                   *  DETERMINE WHETHER THE ACKNOWLEDGEMENT IS VALID.

01124P 0353 A6 05    A RRRNR  LDAA   5,X      FETCH ACKNOWLEDGEMENT
01125P 0355 BD 03CD  P        JSR    CHKNPR   DETERMINE VALID ACKNOWLEDGEMENT
01126P 0358 2B 1F 0379        BMI    IVLRES   INVALID ACKNOWLEDGEMENT IF VALUE IS NEGATIVE


01128                   *  VALID ACKNOWLEDGEMENT.  CLEAR BUFFER, RESET TIMER, RETURN
01129                   *  NODE(S) TO POOL, TEST TO RESTART TIMER, AND UPDATE POINTER.

01131P 035A 6F 00    A        CLR    0,X      DEVAILDATE BUFFER
01132P 035C BD 00AB  P        JSR    RESET1   RESET TIMER
01133P 035F BD 0403  P        JSR    RETURN   RETURN NODE(S)
01134P 0362 B7 17BE  D        STAA   NPROFR   UPDATE ACKNOWLEDGEMENT POINTER
01135P 0365 B1 17BC  D        CMPA   VOFS     COMPARE FOR OUTSTANDING PACKETS
01136P 0368 27 03 036D        BEQ    *+5      OUTSTANDING PACKETS IF VALUE IS NON-ZERO
01137P 036A BD 00BE  P        JSR    START1   START TIMER FOR OUTSTANDING PACKETS
0113BP 036D 39          RT1OUT RTS


01140                   *  THE NODE IS BUSY.  SKIP AROUND THE ABOVE INFORMATION AND
01141                   *  UPDATE THE POINTER (VOFS).  SINCE A PACKET HAS BEEN RECEIVED
01142                   *  THE NODE IS NO LONGER BUSY.  CLEAR THE FLAG.
01143                   *  ALSO, DEVAILDATE THE BUFFER.

01145P 036E A6 05    A RSTVFR LDAA   5,X      FETCH THE ACKNOWLEDGED PACKET NUMBER
01146P 0370 B7 17BC  D        STAA   VOFS     UPDATE THE POINTER
01147P 0373 7F 17BB  D        CLR    NBUSY    READY THE RESPONDER NODE

01149P 0376 6F 00    A SKPRNR CLR    0,X      DEVALIDATE THE BUFFER
0115DP 037B 39                RTS


01152                   *  JUMP TABLE.

01154P 0379 7E 0272  P IVLRES JMP    INVLRS
```

```
01156                    *  REJECT RESPONSE

01158                    *  THIS ROUTINE IS ENTERED WHEN IT IS DETERMINED THAT A REJECT
01159                    *  RESPONSE HAS BEEN RECEIVED.  THE APPROPRIATE ACTION IS TAKEN.

01161                    *  THE POLL BIT IS INTERROGATED FIRST.



01163          037C  P RESREJ EQU    *
01164P 037C F6 179A  0        LDAB   POLLBT   FETCH THE POLL BIT
01165P 037F 27 0E 038F        BEQ    REJNML   NORMAL CONDITION EXISTS IF VALUE IS ZERO


01167                    *  SPECIAL CONDITION EXISTS.  IT HAS TO BE DETERMINED WHETHER
01168                    *  IT IS POSSIBLE TO RETURN TO A NORMAL CONDIION BY CLEARING
01169                    *  THE POLL BIT.  THE ACKNOWLEDGEMENT STATUS WORD IS
01170                    *  COMPARED WITH THE LAST ACKNOWLEDGEMENT WORD BEFORE THE
01171                    *  SEQUENCE WAS INTERRUPTED.  IF THE TWO VALUES MATCH, A
01172                    *  NORMAL CONDITON WILL RE-EXIST.

01174P 0381 E6 05    A        LDAB   5,X      FETCH THE REJECTED ACKNOWLEDGEMENT
01175P 0383 F1 17A6  0        CMPB   XVAR     COMPARE WITH LAST ACKNOWLEDGEMENT
01176P 0386 26 07 038F        BNE    REJNML   NORMAL CONDITON EXISTS IF VALUE IS ZERO
01177P 0388 5F                CLRB           CLEAR THE FOLLOWING:
01178P 0389 F7 1794  0        STAB   TBUSY    TIME OUT BUSY FLAG
01179P 038C F7 179A  0        STAB   POLLBT   INITIATOR POLL BIT


01181                    *  ENTRY POINT FOR NORMAL CONDITON.  SET THE REJECTION FLAG.
01182                    *  DETERMINE WHETHER THE REJECTED PACKET IS A VALID ONE.
01183                    *  RESET (VOFS) AND CLEAR BUFFER.

01185P 038F B6 01    A REJNML LDAA   #1
01186P 0391 B7 17B9  0        STAA   NREJ     SET REJECTION FLAG
01187P 0394 A6 05    A        LDAA   5,X      FETCH THE REJECTED ACKNOWLEDGEMENT
01188P 0396 BD 03C0  P        JSR    CHKNPR   DETERMINE THE VALIDITY OF THE SEQUENCE NUMBER
01189P 0399 2B 0E 0379        BMI    IVLRES   INVALID SEQUENCE NUMBER IF VALUE IS NEGATIVE


01191                    *  REMEMBER NEXT PACKET TO TRANSMIT (VOFS).
01192                    *  UPDATE (VOFS) WITH REJECTED VALUE.

01194P 0398 F6 17BC  0 REJTID LDAB   VOFS     FETCH NEXT SEQUENCE NUMBER
01195P 039E F7 17B6  0        STAB   RNRVAL   SAVE SEQUENCE NUMBER
01196P 03A1 B7 17BC  0        STAA   VOFS     REPLACE SEQUENCE COUNT WITH REJECTED VALUE
01197P 03A4 6F 00    A        CLR    0,X      DEVALIDATE THE BUFFER
01198P 03A6 39                RTS
```

```
01200                   *  RECEIVER NOT READY RESPONSE

01202                   *  THIS ROUTINE IS ENTERED WHEN IT IS DETERMINED THAT THE
01203                   *  RESPONSE IS A RECEIVER NOT READY RESPONSE.  THIS RESPONSE
01204                   *  IS VARY SIMULAR TO THE RR RESPONSE.

01206                   *  THE POLL BIT IS INTERROGATED.



01208          03A7  P RESRNR EQU    *
01209P 03A7 F6 179A  D        LOAB   POLLBT   FETCH THE POLL BIT FOR INTERROGATION
01210P 03AA 27 0E 03BA        BEQ    RNRNM    A NORMAL SITUATION EXISTS IF THE VALUE IS ZERO


01212                   *  A SPECIAL SITUATION EXISTS.  THE LAST ACKNOWLEDGED SEQUENCE
01213                   *  NUMBER BEFORE THE INTERRUPTION IS COMPARED TO THE RECEIVED
01214                   *  SEQUENCE NUMBER.  IF THE VALUES EXIST, A NORMAL SITUATION
01215                   *  IS DECLARED ONCE AGAIN.

01217P 03AC E6 05     A        LOAB   5,X      FETCH ACKNOWLEDGED SEQUENCE NUMBER
01218P 03AE F1 17A6  D        CMPB   XVAR     COMPARE IT TO THE LAST SEQUENCE NUMBER
01219P 03B1 26 07 03BA        BNE    RNRNM    A NORMAL SITUATION EXIST IF THE VALUE IS ZERO


01221                   *  A NORMAL SITUATION IS DECLEARED.  CLEAR OUT THE VARIABLES.

01223P 03B3 5F                CLRB           CLEAR THE FOLLOWING:
01224P 03B4 F7 1794  D        STAB   TBUSY    TIME OUT BUSY FLAG
01225P 03B7 F7 179A  D        STAB   POLLBT   INITIATOR POLL BIT


01227                   *  ENTRY POINT FOR NORMAL SITUATION.  IF NODE IS BUSY
01228                   *  JUMP INTO RR ROUTINE AT APPROPRIATE PLACE.  ELSE, SET
01229                   *  NODE BUSY FLAG, CLEAR REPEAT COUNT, AND JUMP TO RR ROUTINE.

01231P 03BA B6 1788  D RNRNM  LOAA   NBUSY    FETCH NODE BUSY FLAG
01232P 03BD 26 B7 0376        BNE    SKPRNR   NODE IS BUSY IF VALUE IS NON-ZERO
01233P 03BF 7F 17B9  D        CLR    NREJ     CLEAR NODE REJECTION FLAG
01234P 03C2 B6 01     A        LOAA   #1
01235P 03C4 B7 1788  D        STAA   NBUSY    SET NODE BUSY FLAG
01236P 03C7 7F 1793  D        CLR    RCOUNT   CLEAR REPEAT COUNT
01237P 03CA 7E 0353  P        JMP    RRRNR    PASS CONTROL TO RR ROUTINE FOR VERIFICATION, ETC
```

```
01239                    *  VERIFY ACKNOWLEDGEMENT

01241                    *  THIS ROUTINE IS CALLED BY THE RECEIVER SECTION OF THE INITIATOR
01242                    *  TO CHECK OUT THE VALIDITY OF THE RECEIVED SEQUENCE NUMBER.
01243                    *  THE ROUTINE (COMPAR) PERFORMS THE VALIDITY CHECK.  THIS ROUTINE
01244                    *  SETS UP THE VARIABLES.  THE TEMPORARY LOCATION (TEMPR2) IS USED.
D1245                    *  THE ROUTINE WILL SET UP THE POINTER (VOFS) IN THE LSB AND (NPROFR)
01246                   *  IN THE MSB OF THE VARIABLE (TEMPR2).  THE ROUTINE (COMPAR) IS
 01247                   *  CALLED.

 01249          03CD  P CHKNPR EQU    *
 01250P 03CD F6 178E  D        LDAB   NPROFR   FETCH THE ACKNOWLEDGEMENT POINTER
 01251P 03D0 F7 176E  D        STAB   TEMPR2   INITIALIZE THE MSB
 01252P 03D3 F6 178C  D        LDAB   VOFS     FETCH THE SEQUENCE POINTER
 01253P 03D6 F7 176F  D        STAB   TEMPR2+1 INITIALIZE THE LSB
 01254P 03D9 BD 03DD  P        JSR    COMPAR   CHECK THE VALIDITY
 01255P 03DC 39                RTS
```

```
01257                    *  COMPARE

01259                    *  THE PURPOSE OF THIS ROUTINE IS TO DETERMINE WHETER A GIVEN
01260                    *  NUMBER IS BETWEEN TWO OTHER NUMBERS.  THE OTHER TWO NUMBERS
01261                    *  ARE GIVEN IN (TEMPR2).  THE LSB HOLDS THE GREATER VALUE AND
01262                    *  THE MSB HOLDS THE LESSER VALUE.  THE NUMBER IN QUESTION IS
01263                    *  PASSED IN ACCUMULATOR A.

01265                    *  FIRST, COMPARE THE TWO VALUES AND
01266                    *  DETERMINE WHICH IS GREATER.



0126B          0300  P COMPAR EQU    *
01269P 0300 F6 176E  0        LOAB   TEMPR2   FETCH THE LOWER VALUE
0127OP 03E0 F1 176F  0        CMPB   TEMPR2+1 COMPARE IT WITH THE OTHER VALUE
01271P 03E3 20 0E 03F3        BLT    OTHER    SITUATION REVERSED IF BRANCH IS NOT TAKEN


01273                    *  THE SITUATION IS:
01274                    *          (TEMPR2) ¶ (TEMPR2+1)

01276P 03E5 11                CBA             COMPARE VALUE IN QUESTION WITH MSB
01277P 03E6 2C 16 03FE        BGE    VALID    VALID IF GREATER THAN

01279P 03EB F6 176F  0        LOAB   TEMPR2+1 FETCH LSB
012B0P 03EB 11                CBA             COMPARE VALUE IN QUESTION WITH LSB
012B1P 03EC 20 10 03FE        BLT    VALID    VALID NUMBER IF LESS THAN LSB
012B2P 03EE 27 10 0400        BEQ    EVALID   VALID NUMBER AND EQUAL TO LSB

012B4P 03F0 C6 50    A NVALID LOAB   #B0      INVALID NUMBER - ISSUE CODE
012B5P 03F2 39                RTS


012B7P 03F3 11         OTHER  CBA             COMPARE VALUE IN QUESTION WITH MSB
012B8P 03F4 20 FA 03F0        BLT    NVALID   INVALID NUMBER IF IT IS LESS THAN MSB

01290P 03F6 F6 176F  0        LOAB   TEMPR2+1 FETCH THE LSB
01291P 03F9 11                CBA             COMARE VALUE WITH THE LSB
01292P 03FA 2E F4 03F0        BGT    NVALID   NUMBER IS INVALID IF GREATER THAN LSB
01293P 03FC 27 02 0400        BEQ    EVALID   NUMBER IS EQUAL TO THE LSB

01295P 03FE 5F         VALID  CLRB            VALID NUMBER - INDICATE SUCH
01296P 03FF 39                RTS

0129BP 0400 C6 01    A EVALID LOAB   #$01     VALID AND EQUAL NUMBER
01299P 0402 39                RTS
```

```
01301                   *  RETURN A NODE

01303                   *  THIS ROUTINE WILL RETURN ALL OUTSTANDING NODES BETWEEN THE
01304                   *  VARIABLES (NPROFR) AND THE RECEIVED SEQUENCE NUMBER  LOCATED
01305                   *  IN ACCUMULATOR A.


01307          0403  P RETURN EQU    *
0130BP 0403 B1 17BE  0         CMPA   NPROFR   CHECK FOR POSSIBLE RETURNS
01309P 0406 27 21 0429         BEQ    RETENO   NO RETURNS MADE IF VALUE IS ZERO
0131OP 040B 36                 PSHA            SAVE ACCUMULATOR A FOR EXITING

01312                   *  SUBTRACT ONE FROM GIVEN NUMBER BY ADDING SEVEN MODULO B.

01314P 0409 C6 07    A         LDAB   #7
01315P 040B BD 0101  P         JSR    MODULO   SUBTRACT ONE
01316P 040E 16                 TAB             TRANSFER DESTINATION VALUE
01317P 040F B6 17BE  0         LDAA   NPROFR   FETCH SOURCE VALUE
0131BP 0412 20 07 041B         BRA    RETSTR   RETURN FIRST NODE

01320                   *  INCREMENT SOURCE ADDRESS BY ONE USING MODULO 8 FORMAT

01322P 0414 37           RETUPO PSHB
01323P 0415 C6 01    A         LDAB   #1
01324P 0417 BD 0101  P         JSR    MODULO   ADD ONE TO SOURCE POINTER
01325P 041A 33                 PULB
01326P 041B 36           RETSTR PSHA            SAVE SOURCE VALUE
01327P 041C BD 01B7  P         JSR    FINDTB   FIND PHYSICAL ADDRESS WITHIN (INTABL)
0132BP 041F EE 01    A         LDX    1,X      FETCH NODE ADDRESS FROM TABLE
01329P 0421 BD 0076  P         JSR    PUTNOD   RETURN NODE TO THE NODE POOL
01330P 0424 32                 PULA            RETURN SOURCE VALUE TO ACC A
01331P 0425 11                 CBA             COMPARE SOURCE WITH DESTINATION
01332P 0426 26 EC 0414         BNE    RETUPO   MORE NODES TO RETURN IF VALUE IS NON-ZERO
01333P 0428 32                 PULA
01334P 0429 39           RETENO RTS
```

```
01336                   *  TIME OUT

0133B                   *  THE TIMER IS INITIALIZED AND STARTED FOR EACH PACKET THAT IS
01339                   *  BEING TRANSMITTED.  THE TIMER IS STOPPED AND CLEARED FOR EACH
0134O                   *  CORRECT RESPONSE IT RECEIVES.  IF A TIME OUT OCCURS, THE
01341                   *  INITIATOR HAS NOT RECEIVED THE APPROPRIATE RESPONSE FOR ITS
01342                   *  TRANSMITTED COMMAND PACKET.  A TIME OUT WILL PASS CONTROL TO
01343                   *  THIS ROUTINE.  THIS ROUTINE WILL BACK UP AND RETRANSMIT
01344                   *  THE UNACKNOWLEDGED PACKETS.  THIS WILL OCCUR FOR (N2NUM) TIMES.
01345                   *  AFTER THAT TIME, A RESPONSE REJECTION COMMAND WILL BE SENT.
01346                   *  THIS IS EQUIVALENT TO AN UNRECOVERABLE ERROR.



0134B          042A  P T1EXPR EQU    *
01349P 042A 7F 17A5  D        CLR    TIME1    CLEAR TIME OUT FLAG

01351                   *  FETCH THE BUSY FLAG AND DETERMINE WHETHER THIS IS THE FIRST
01352                   *  TIME FOR THE TIME OUT (INITIAL) OR WHETHER THIS IS A
01353                   *  REPEATED TIME OUT CONDITION.

01355P 042D B6 1794  D        LDAA   TBUSY    FETCH BUSY FLAG
01356P 0430 26 09 043B        BNE    AREDY    INITIAL TIME OUT IF VALUE IS ZERO
01357P 0432 7C 1794  D        INC    TBUSY    SET TIME OUT FLAG

01359                   *  BACK UP THE SEQUENCE COUNTER (VOFS).  BUT
0136O                   *  FIRST, SAVE THE OLO VALUE OF IT.

01362P 0435 B6 17BC  D        LDAA   VOFS     FETCH THE SEQUENCE POINTER
01363P 043B B7 17A6  D        STAA   XVAR     SAVE THE OLD VALUE


01365                   *  ENTRY POINT FOR REPEATED TIME OUT

01367P 043B F6 17BE  D AREDY  LDAB   NPROFR   FETCH THE NEW VALUE OF (VOFS)
0136BP 043E F7 17BC  D        STAB   VOFS     BACK UP THE SEQUENCE POINTER
01369P 0441 7C 1793  D        INC    RCOUNT   INCREMENT THE REPEAT COUNT BY ONE
0137OP 0444 B6 1793  D        LDAA   RCOUNT   FETCH THE REPEAT COUNT JUST INCREMENTED
01371P 0447 B1 04    A        CMPA   #N2NUM   COMPARE WITH THE MAXIMUM REPEAT COUNT
01372P 0449 27 06 0451        BEQ    IIVLRS   EXCEEDED ITS LIMIT IF VALUE IS ZERO
01373P 044B B6 01    A        LDAA   #1
01374P 044D B7 179A  D        STAA   POLLBT   SET POLL BIT AND TRY AGAIN
01375P 0450 39                RTS


01377                   *  JUMP TABLE.

01379P 0451 7E 0272  P IIVLRS JMP    INVLRS
```

```
01381                  ************************************************************************
01382                  *                                                                      *
01383                  *    RESPONDER                                                         *
01384                  *                                                                      *
01385                  *    THE RESPONDER RECEIVES COMMAND PACKETS AND RESPONDS TO THEM       *
01386                  *    ACCORDINGLY.  THERE ARE FOUR RESPONDES SIMULATED HERE; THEY       *
01387                  *    ARE THE RR, RNR, REJ, AND UA RESPONSES.  DURING A NORMAL          *
01388                  *    TRANSMISSION, THE RR RESPONSE IS USED EXTENSIVELY.  WHEN AN       *
01389                  *    ABNORMAL CONDITION EXISTS, THE OTHER RESPONSES ARE USED TO AID    *
01390                  *    IN THE RECOVERY ACTION.                                           *
01391                  *                                                                      *
01392                  *    THE RESPONDER RECEIVES PACKETS OF INFORMATION FROM THE RECEIVER   *
01393                  *    THROUGH THE IPC BUFFER.  IN TERN, THE RESPONDES ARE               *
01394                  *    ISSUED IN ACCORDANCE WITH THE STATUS OF THE INCOMING PACKET.      *
01395                  *    THE PACKET ADDRESSES ARE QUEUED IN THE (OUTABL) TABLE.  THEY      *
01396                  *    ARE DEQUEUED AND SENT TO THE USER (OUTPUT ROUTINE) WHEN           *
01397                  *    REQUESTED.  THIS IS ACCOMPLISHED THROUGH THE (OUTDAT) IPC         *
01398                  *    BUFFER.                                                           *
01399                  *                                                                      *
01400                  ************************************************************************



01402          0000  A        IFEQ   CHANGE   ASSEMBLE FOR THE (DTE) IF THE VALUE IS ZERO
01403          0454  P TKPC1  EQU    *
01404                         ENDC

01406          0000  A        IFNE   CHANGE   ASSEMBLE FOR (DCE) IF THE VALUE IS NON-ZERO
01407                  TKPC8 EQU *
01408                    ENDC


01410                  *  NORMAL LOOP

01412                  *  THE NORMAL RESPONDER LOOP POLLS TWO BUFFERS.  THEY ARE THE
01413                  *  (OUTDAT) AND THE (INCOMD).  RESPONDES ARE ISSUED TO THE (OUTRES)
01414                  *  BUFFER.  AT THE END OF THE LOOP, CONTROL IS PASSED
01415                  *  TO THE SCHEDULER.

01417P 0454 FE 1766  D RNORML LDX    OUTDAT   FETCH THE PRIMARY ADDRESS
0141BP 0457 A6 00    A        LDAA   0,X      TEST THE STATUS WORD
01419P 0459 26 16 0471        BNE    RSKIP2   BUFFER EMPTY IF VALUE IS ZERO
```

```
01421                    *  BUFFER IS EMPTY, CHECK TO SEE IF THERE
01422                    *  IS SOMETHING TO DEQUEUE FROM (OUTABL).

01424P 045B B6 1792  D RSKIP1 LDAA   VALIDF   FETCH FRONT POINTER OF (OUTABL)
01425P 045E B1 1795  D        CMPA   VALIDR   COMPARE FRONT WITH REAR POINTERS
01426P 0461 27 0E 0471        BEQ    RSKIP2   QUEUE IS EMPTY IF VALUE IS ZERO
01427P 0463 BD 049C  P        JSR    RSNDAT   DEQUEUE A DATA PACKET AND ISSUE IT


01429                    *  INCREASE THE FRONT POINTER BY ONE.

01431P 0466 B6 1792  D        LDAA   VALIDF   FETCH FRONT POINTER
01432P 0469 C6 01    A        LDAB   #1
01433P 046B BD 04BE  P        JSR    MODL02   INCREMENT FRONT POINTER BY ONE
01434P 046E B7 1792  D        STAA   VALIDF   UPDATE FRONT POINTER


01436                    *  IN COMMAND

01438                    *  THE SECONARY BUFFER OF (INCOMD) IS INTERROGATED.  IF THE
01439                    *  STATUS WORD IS VALID,  THE CHECK OUT ROUTINE (CHKOUT) IS
01440                    *  CALLED.  OTHERWISE, THE PRIMARY BUFFER IS SCANNED.  IF
01441                    *  IT IS VALID, THE BUFFERS ARE SWAPPED AND THE (CHKOUT)
01442                    *  ROUTINE IS CALLED.

01444P 0471 FE 1764  D RSKIP2 LDX    STDBY5   FETCH THE SECONDARY POINTER
01445P 0474 A6 00    A        LDAA   0,X      INTERROGATE STATUS WORD
01446P 0476 26 1E 0496        BNE    RSKIP5   BUFFER IS VALID IF VALUE IS NON-ZERO

01448P 0478 FE 1762  D        LDX    INCOMD   FETCH THE PRIMARY BUFFER POINTER
01449P 047B A6 00    A        LDAA   0,X      INTERROGATE THE STATUS WORD
01450P 047D 27 1A 0499        BEQ    ISKIP3   BUFFER IS VALID IF THE VALUE IS NON-ZERO

01452P 047F 0F                SEI
01453P 0480 7C 0000  A        INC    CRTFLG   ENTER CRITICAL REGION
01454P 0483 FF 1774  D        STX    TEMPR5
01455P 0486 FE 1764  D        LDX    STDBY5
01456P 0489 FF 1762  D        STX    INCOMD
01457P 048C FE 1774  D        LDX    TEMPR5
01458P 048F FF 1764  D        STX    STDBY5
01459P 0492 7A 0000  A        DEC    CRTFLG   EXIT CRITICAL REGION
01460P 0495 0E                CLI
01461P 0496 BD 0504  P RSKIP5 JSR    CHKOUT   CALL CHECK OUT ROUTINE


01463                    *  RETURN CONTROL TO THE SCHEDULER.  ON THE NEXT PASS,
01464                    *  RETURN CONTROL TO THE TOP OF THE NORAL RESPONDER LOOP.

01466P 0499 3F         ISKIP3 SWI
01467P 049A 20 B8 0454        BRA    RNORML   REPEAT LOOP
```

```
01469                    *  DEQUEUE DATA

01471                    *  IF THERE IS INFORMATION IN THE OUTPUT QUEUE AND THE IPC OUTPUT
01472                    *  BUFFER IS EMPTY, THIS ROUTINE WILL GAIN CONTROL.  THE FRONT
01473                    *  POINTER (VALIDF) IS USED TO LOCATE THE PACKET WITHIN THE
01474                    *  QUEUE.  THE PRIMARY POINTER IS FETCHED AND THE BUFFER
01475                    *  IS VALIDATED.  IF THE HOST BUSY FLAG IS SET, THE ROUTINE
01476                    *  WILL CALL (CLRBSY).



0147B          049C  P RSNDAT EQU    *
01479P 049C B6 1792  D        LDAA   VALIDF   FETCH THE FRONT POINTER OF (OUTABL)
014B0P 049F BD 04C6  P        JSR    FNDTB2   CALCULATE THE PHYSICAL ADDRESS INTO (OUTABL)
014B1P 04A2 A6 00    A        LDAA   0,X      RETRIEVE THE BYTE COUNT
014B2P 04A4 36                PSHA
014B3P 04A5 A6 02    A        LDAA   2,X      RETRIEVE THE LSB OF THE DATA PACKET
014B4P 04A7 E6 01    A        LDAB   1,X      RETRIEVE THE MSB OF THE DATA PACKET

014B6P 04A9 FE 1766  D        LDX    OUTDAT   FETCH THE PRIMARY BUFFER POINTER
014B7P 04AC E7 02    A        STAB   2,X      INITIALIZE BUFFER WITH THE MSB OF THE DATA PACKET
014B8P 04AE A7 03    A        STAA   3,X      INITIALIZE BUFFER WITH THE LSB OF THE DATA PACKET
014B9P 04B0 32                PULA
01490P 04B1 A7 01    A        STAA   1,X      INITIALIZE THE BUFFER WITH THE PACKET TYPE
01491P 04B3 A7 00    A        STAA   0,X      VALIDATE THE BUFFER

01493P 04B5 F6 17BA  D        LDAB   HBUSY    INTERROGATE THE HOST BUSY FLAG
01494P 04BB 27 03 04BD        BEQ    *+5      FLAG IS SET IF VALUE IS NON-ZERO
01495P 04BA BD 064B  P        JSR    CLRBSY   ENTER CLEAR BUSY ROUTINE
01496P 04BD 39                RTS
```

```
0149B                   *  MODULO EIGHT

01500                   *  THIS ROUTINE WILL ADD TWO NUMBERS USING MODULO 8 FORMAT.
01501                   *  THE NUMBERS ARE PASSED IN ACCUMULATORS A AND B AND
01502                   *  THE RESULT IS CONTAINED IN ACCUMULATOR A.

01504           04BE  P MODL02 EQU    *
01505P 04BE 1B                  ABA             CONVENTIONAL ADD
01506P 04BF B1 07     A         CMPA   #7       COMPARE ADDITION WITH UPPER LIMIT
01507P 04C1 2F 02 04C5          BLE    *+4      ADJUSTMENT MADE IF BRANCH NOT TAKEN
01508P 04C3 B0 08     A         SUBA   #8       ADJUST FOR MODULO 8 ADDITION
01509P 04C5 39                  RTS



01511                   *  FIND PHYSICAL ADDRESS

01513                   *  THIS ROUTINE WILL CALCULATE THE PHYSICAL ADDRESS WHICH WILL
01514                   *  POINT INTO THE (OUTABL) TABLE FROM A GIVEN POINTER.  THE
01515                   *  POINTER IS PASSED IN ACCUMULATOR A AND THE RETURNING
01516                   *  ADDRESS IS CONTAINED IN THE INDEX REGISTER.


0151B           04C6  P FNDTB2 EQU    *
01519P 04C6 37                  PSHB            SAVE ACC B
01520P 04C7 CE 173A   D         LDX    #OUTABL  FETCH THE BASE ADDRESS OF THE TABLE
01521P 04CA FF 1774   D         STX    TEMPR5   STORE ADDRESS IN WORKING REGISTER
01522P 04CD 16                  TAB             MULTIPLY ACC A BY THREE
01523P 04CE 48                  ASLA
01524P 04CF 1B                  ABA
01525P 04D0 BB 1775   D         ADDA   TEMPR5+1 ADD DISPLACEMENT TO LSB OF BASE ADDRESS
01526P 04D3 B7 1775   D         STAA   TEMPR5+1 UPDATE BASE ADDRESS
01527P 04D6 24 03 04DB          BCC    *+5      TEST FOR OVERFLDW
01528P 04DB 7C 1774   D         INC    TEMPR5   COMPENSATE FOR CARRY
01529P 04DB FE 1774   D         LDX    TEMPR5   FETCH NEWLY CALCULATED ADDRESS
01530P 04DE 33                  PULB            RETRUN ACC B
01531P 04DF 39                  RTS
```

```
01533                    *  QUEUE DATA

01535                    *  THIS ROUTINE WILL RETRIEVE THE NECESSARY INFORMATION FROM
01536                    *  AN IPC BUFFER AND WILL QUEUE IT INTO THE (OUTABL) TABLE.
01537                    *  THE BUFFER IS DEVALIDATED AND THE REAR POINTER (VALIDR)
0153B                    *  IS INCREMENTED BY ONE.  IT IS ASSUMED THAT THE BUFFER
01539                    *  POINTER IS CONTAINED IN THE INDEX REGISTER.


01541          04E0  P RMOVIT EQU    *
01542P 04E0 A6 02    A        LDAA   2,X      RETRIEVE THE BYTE COUNT
01543P D4E2 36                PSHA
01544P 04E3 E6 06    A        LDAB   6,X      RETRIEVE THE MSB OF THE DATA PACKET
01545P 04E5 A6 07    A        LDAA   7,X      RETRIEVE THE LSB OF THE DATA PACKET
01546P 04E7 36                PSHA
D1547P 04EB 6F 00    A        CLR    0,X      DEVALIDATE THE BUFFER

D1549P 04EA B6 1795  D        LDAA   VALIDR   FETCH THE REAR POINTER
01550P 04ED BD 04C6  P        JSR    FNDTB2   CALCULATE THE PHYSICAL ADDRESS
01551P D4F0 32                PULA
01552P D4F1 A7 02    A        STAA   2,X      INITIALIZE THE QUEUE WITH THE LSB
D1553P 04F3 E7 01    A        STAB   1,X      INITIALIZE THE QUEUE WITH THE MSB
01554P 04F5 32                PULA
D1555P 04F6 A7 00    A        STAA   0,X      INITIALIZE THE QUEUE WITH THE BYTE COUNT

01557                    *  INCREMENT THE REAR POINTER BY ONE

01559P 04FB B6 1795  D        LDAA   VALIDR   FETCH THE REAR POINTER
D1560P 04FB C6 01    A        LDAB   #1
01561P 04FD BD 04BE  P        JSR    MODL02   INCREMENT THE REAR POINTER BY ONE
D1562P 0500 B7 1795  D        STAA   VALIDR   UPDATE THE REAR POINTER
01563P 0503 39                RTS
```

```
01565                    *  CHECK OUT

01567                    *  THIS ROUTINE IS CALLED BY THE RECEIVING SIDE OF THE INITIATOR
01568                    *  TO CHECK OUT THE INCOMING PACKET.  THE PACKET IS INTERROGATED
01569                    *  FOR VALIDITY.  IF THE TEST PASSES, FURTHER INTERROGATION CONTINUES.
01570                    *  FINALLY, AND ALL GOING WELL, THE PACKET IS ASSUMED TO BE A RR
01571                    *  RESPONSE.  APPROPRIATE ACTION IS TAKEN.  IT IS FURTHER ASSUMED
01572                    *  THAT THE SECONARY POINTER IS PASSED IN THE INDEX REGISTER.



01574           0504  P CHKOUT EQU    *
01575P 0504 A6 01     A        LDAA   1,X       INTERROGATE STATUS WORD
01576P 0506 2B 66 056E         BMI    ISREJ     INVALID STATUS WORD IF VALUE IS NEGATIVE
01577P 0508 81 02     A        CMPA   #2        TEST FOR AN SARM COMMAND
01578P 050A 27 68 0574         BEQ    RSARM     ILLEGAL RESPONSE IF VALUE IS ZERO
01579P 050C 81 03     A        CMPA   #3        TEST FOR A DISCONNECT COMMAND
01580P 050E 27 58 0568         BEQ    ISDSC     ILLEGAL RESPONSE IF VALUE IS ZERO
01581P 0510 2E 59 0568         BGT    ICMDR     ISSUE COMMAND REJECT IF VALUE IS GREATER THAN ZERO


01583                    *  IF THE HOST IS BUSY, SKIP AROUND ALL OF THIS AND IGNORE THE
01584                    *  PACKET OF INFORMATION.  OTHERWISE, CHECK FOR THE REJECT FLAG
01585                    *  TO BE SET.  SKIP AROUND THE FINAL BIT TEST IF THE FLAG IS SET.
01586                    *  THIS IS FOR THE RESPONDER TO REJECT PACKETS OF INFORMATION
01587                    *  AFTER A START UP AND BEFORE THE INITIATOR HAS A CHANCE TO
01588                    *  TRANSMIT THE CORRECT PACKET WITH ITS CORRECT POLL BIT SET.

01590P 0512 B6 178A   0        LDAA   HBUSY     INTERROGATE HOST BUSY FLAG
01591P 0515 26 46 0550         BNE    HEXIT     IGNORE PACKET IF BUSY FLAG IS SET

01593P 0517 B6 1788   0        LDAA   HREJ      INTERROGATE THE HOST REJECT FLAG
01594P 051A 26 13 052F         BNE    HCT1      HOST REJECTION STATE IF VALUE IS NON-ZERO


01596                    *  IF THE POLL BIT IS CLEAR, SKIP AROUND.
01597                    *  OTHERWISE, SET THE FINAL BIT TO ONE.

01599P 051C A6 03     A        LDAA   3,X       INTERROGATE THE RECEIVED POLL BIT
01600P 051E 27 07 0527         BEQ    HCNT1     POLL BIT IS CLEAR IF VALUE IS ZERO
01601P 0520 B6 01     A        LDAA   #1
01602P 0522 B7 1799   0        STAA   FINBIT    SET THE FINAL BIT ACCORDINGLY
01603P 0525 20 08 052F         BRA    HCT1      RETURN TO THE NORMAL FLOW
```

```
01605                   *  ENTRY POINT FOR NORMAL FLOW.  CLEAR THE FINAL BIT SINCE THE
01606                   *  INCOMING POLL BIT IS ZERO.  TEST THE (HCBUSY) FLAG.  THIS
01607                   *  FLAG IS SET AFTER A RESTART IS IN EFFECT AND BEFORE THE SYSTEM
01608                   *  DECLARES A NORMAL FLOW ONCE AGAIN.  IF SET, REJECT THE
01609                   *  PACKETS UNTIL THE SEQUENCE NUMBERS MATCH.

01611P 0527 7F 1799  0 HCNT1  CLR    FINBIT   CLEAR FINAL BIT
01612P 052A B6 1796  0 HCCNTU LDAA   HCBUSY   INTERROGATE THE HOST BUSY INTERMEDIATE FLAG
01613P 052D 27 0A 0539        BEQ    HCNTRL   NORMAL CONDITION IF VALUE IS ZERO
01614P 052F B6 17B0  0 HCT1   LDAA   VOFR     FETCH THE LAST CORRECT SEQUENCE NUMBER
01615P 0532 A1 04    A        CMPA   4,X      COMPARE WITH INCOMING SEQUENCE NUMBER
01616P 0534 26 27 055D        BNE    HEXIT    NORMAL CCONDITION DELCARED IF VALUE IS ZERO
01617P 0536 7F 1796  0        CLR    HCBUSY   DECLARE NORMAL CONDITION


01619                   *  NORMAL SEQUENCE CHECK.  THIS IS A NORMAL CHECK MADE TO DETERMINE
01620                   *  WHICH NUMBER THE INCOMING PACKET HAS THE CORRECT SEQUENCE
01621                   *  IF NOT, REJECT THE RESPONSE.  AT THIS POINT, A CORRECT SEQUENCE
01622                   *  NUMBER SHOULD BE RECEIVED AND ANY EXTRA PACKETS WOULD HAVE
01623                   *  PACKETS WOULD HAVE BEEN ELIMINATED.

01625P 0539 B6 178D  0 HCNTRL LDAA   VOFR     FETCH THE CURRENT SEQUENCE NUMBER
01626P 053C A1 04    A        CMPA   4,X      COMPARE WITH THE INCOMING SEQUENCE NUMBER
01627P 053E 26 2E 056E        BNE    ISREJ    PACKETS ARE IN ORDER IF VALUE IS ZERO
```

```
01629                   *  IT IS ASSUMED THAT THE PACKET RECEIVED IS A VALID ONE.  NOW,
01630                   *  THE SIMPLE BOOK WORK PROCEDURE OF ACKNOWLEDGING THE PACKET
01631                   *  TRANSMITTED AND ISSUING THE CORRECT RESPONSE PACKET IS DONE.

01633                   *  INCREMENT THE NEXT SEQUENCE NUMBER POINTER (VOFR).

01635P 0540 B6 17BD  D        LDAA   VOFR     FETCH THE NEXT VALID SEQUENCE POINTER
01636P 0543 C6 01    A        LDAB   #1
01637P 0545 BD 04BE  P        JSR    MODLO2   INCREMENT THE SEQUENCE NUMBER BY ONE MODULO B
0163BP 054B B7 17BD  D        STAA   VOFR     UPDATE THE SEQUENCE POINTER
01639P 054B 7F 17BB  D        CLR    HREJ     CLEAR REJECTION FLAG
0164OP 054E BD 04EO  P        JSR    RMOVIT   QUEUE THE DATA
0164lP 0551 BD 04BE  P        JSR    MODLO2   INCREMENT THE REAR POINTER BY ONE


01643                   *  THE DATA PACKET IS QUEUED WAITING FOR OUTPUT.  THE CORRECT
01644                   *  RESPONSE ACKNOWLEDGEMENT IS DETERMINED.  IF THE QUEUE IS FULL
01645                   *  AN RNR RESPONSE WILL BE ISSUED.  OTHERWISE, AN RR RESPONSE WILL
01646                   *  BE SENT.

0164BP 0554 B1 1792  D        CMPA   VALIDF   COMPARE FROMT POINTER WITH REAR POINTER
01649P 0557 27 1B 0571        BEQ    ISRNR    ISSUE AN RNR RESPONSE IF QUEUE IS FULL
0165OP 0559 BD 0611  P        JSR    ISSRR    ISSUE AN RR RESPONSE
0165lP 055C 39                RTS


01653                   *  THIS ROUTINE IS USED TO IGNORE THE INCOMING DATA PACKETS.  THE
01654                   *  NODE IS RETURNED TO THE NODE POOL AND THE BUFFER IS DEVALIDATED.

01656P 055D EE 06    A HEXIT  LDX    6,X      RETRIEVE THE NODE ADDRESS
01657P 055F BD 0076  P        JSR    PUTNOD   RETURN THE NODE TO THE POOL
0165BP 0562 FE 1764  D        LDX    STDBY5   FETCH THE SECONDARY POINTER
01659P 0565 6F 00    A        CLR    0,X      DEVALIDATE THE BUFFER
0166OP 0567 39                RTS


01662                   *  JUMP TABLE.

01664P 056B 7E 05A4  P ISDSC  JMP    ISSDSC   RECEIVED A DISCONNECT COMMAND
01665P 056B 7E 05C6  P ICMDR  JMP    ISCMDR   REJECT THE RESPONSE
01666P 056E 7E 0626  P ISREJ  JMP    ISSREJ   ISSUE REJECTION RESPONSE
01667P 0571 7E 066C  P ISRNR  JMP    ISSRNR   ISSUE AN RNR RESPONSE
```

```
01669                    *  RESTART RESPONDER

01671                    *  THIS ROUTINE WILL RESTART OR RESET THE RESPONDER.  THE OUTPUT
01672                    *  QUEUE IS FLUSHED AND THE OUTSTANDING NODES ARE RETURNED TO
01673                    *  THE NODE POOL.  THE VARIABLES ARE CLEARED AND THE RECEIVER
01674                    *  ON LINE INDICATOR IS SET.



01676           0574  P RSARM  EQU    *
01677P 0574 BD 05AF  P         JSR    RXMTUA   ISSUE AN UNNUMBERED ACKNOWLEDGEMENT - UA


01679                    *  RETURN NODES TO NODE POOL.

01681P 0577 F6 1795  0 RLOOP  LDAB   VALIOR   FETCH REAR POINTER
01682P 057A F1 1792  0        CMPB   VALIOF   COMPARE FRONT AND REAR POINTERS
01683P 057D 27 13 0592        BEQ    RCLEAR   QUEUE IS EMPTY IF VALUE IS ZERO
01684P 057F 17                TBA
01685P 0580 BD 04C6  P        JSR    FNDTB2   FIND PHYICAL ADDRESS INTO (OUTABL) TABLE
01686P 0583 EE 01    A        LDX    1,X      FETCH NODE ADDRESS FROM TABLE
01687P 0585 BD 0076  P        JSR    PUTNOD   RETURN NODE TO POOL
01688P 0588 B6 01    A        LDAA   #1
01689P 058A BD 04BE  P        JSR    MODLO2   INCREMENT THE REAR POINTER BY ONE
01690P 058D B7 1795  0        STAA   VALIOR   UPDATE THE REAR POINTER
01691P 0590 20 E5 0577        BRA    RLOOP    REPEAT THE LOOP


01693                    *  FLUSH OUT THE BUFFERS BY CLEARING OUT THE VARIABLES.

01695P 0592 4F          RCLEAR CLRA           CLEAR THE FOLLOWING VARIABLES:
01696P 0593 B7 17BD  0        STAA   VOFR     NEXT VALID SEQUENCE NUMBER
01697P 0596 B7 1795  0        STAA   VALIOR   OUTPUT TABLE REAR POINTER
01698P 0599 B7 1792  0        STAA   VALIOF   OUTPUT TABLE FRONT POINTER
01699P 059C B7 1799  0        STAA   FINBIT   RESPONDER FINAL BIT
01700P 059F 4C                INCA
01701P 05A0 B7 17A2  0        STAA   HONLNR   SET THE HOST ON LINE RECEIVER INDICATOR
01702P 05A3 39                RTS
```

```
01704                   *  ISSUE DISCONNECT

01706                   *  THIS ROUTINE WILL BE CALLEO WHEN AN (SARM) COMMAND IS RECEIVED.
01707                   *  THIS ROUTINE WILL ISSUE AN ACKNOWLEOGEMENT VIA THE (SARM)
01708                   *  ROUTINE.  THE HOST ON LINE INOICATOR IS CLEAREO INDICATING
01709                   *  THAT THE LINK IS DOWN.  CONTROL IS PASSED TO THE SCHEDULER.
01710                   *  ONCE TIS ROUTINE RECEIVES CONTROL, IT IS PASSED TO THE NORMAL
01711                   *  EXECUTION LOOP.

01713          05A4  P ISSDSC EQU    *
01714P 05A4 BD 0574  P         JSR    RSARM    ISSUE ACKNOWLEOGEMENT
01715P 05A7 4F               CLRA
01716P 05AB B7 17A2  D         STAA   HONLNR   CLEAR HOST ON LINE INDICATOR
01717P 05AB 3F               SWI             RETURN CONTROL TO SCHEOULER
01718P 05AC 7E 0454  P         JMP    RNORML   RETURN CONTROL TO THE NORMAL EXECUTION LOOP



01720                   *  ISSUE AN UA RESPONSE

01722                   *  THIS ROUTINE WILL ISSUE AN UNNUMBEREO ACKNOWLEDGEMENT.  THE
01723                   *  ROUTINE WILL WAIT FOR THE PRIMARY BUFFER TO BE EMPTY AND WHEN
01724                   *  IT IS, INITIALIES THE BUFFER FOR AN UA RESPONSE.


01726          05AF  P RXMTUA EQU    *
01727P 05AF 7F 1799  D         CLR    FINBIT   CLEAR THE FINAL BIT
01728P 05B2 6D 03    A         TST    3,X      TEST THE RECEIVEO COMMAND POLL BIT
01729P 05B4 27 03 05B9         BEQ    *+5      FINAL BIT IS ZERO IF VALUE IS ZERO
01730P 05B6 7C 1799  D         INC    FINBIT   SET FINAL BIT
01731P 05B9 6F 00    A         CLR    0,X      OEVALIOATE THE COMMAND SECONARY BUFER
01732P 05BB CE 175E  D         LOX    #OUTRES  FETCH THE OUTPUT RESPONSE PRIMARY BUFFER
01733P 05BE BD 00B0  P         JSR    WAITBF   WAIT FOR THE BUFFER TO BECOME EMPTY
01734P 05C1 C6 07    A         LDAB   #7       LOAO WITH THE U.A. RESPONSE CODE
01735P 05C3 7E 065B  P         JMP    RLOAD    ISSUE THE RESPONSE
```

```
01737                    *  ISSUE COMMAND REJECT

01739                    *  THIS ROUTINE WILL ISSUE A COMMAND REJECT RESPONSE.  THE ROUTINE
01740                    *  WILL WAIT UNTIL A NEW COMMAND IS RECEIVED BEFORE EXITING.
01741                    *  IT WILL WAIT FOR A NEW COMMAND THROUGH THE ROUTINE (RWAIT).


01743          05C6  P ISCMOR EQU    *
01744P 05C6 CE 175E  0        LDX    #OUTRES  FETCH PRIMARY POINTER
01745P 05C9 BD 00BD  P        JSR    WAITBF   WAIT FOR BUFFER TO BECOME INVALID
01746P 05CC B6 0B    A        LDAA   #B       LOAD WITH COMMAND REJECT CODE
01747P 05CE BD 065B  P        JSR    RLOAD    ISSUE THE RESPONSE

01749P 05D1 BD 05EA  P        JSR    RWAIT    WAIT FOR A COMMAND
01750P 05D4 A6 01    A        LDAA   1,X      FETCH THE PACKET TYPE
01751P 05D6 27 4E 0626        BEQ    ISSREJ   REJECT COMMAND IF VALUE IS ZERO

01753P 05DB A6 03    A        LDAA   3,X      FETCH FINAL BIT
01754P 05DA B7 1799  0        STAA   FINBIT   SET FINAL BIT ACCORDINGLY
01755P 05DD A6 01    A        LDAA   1,X      FETCH PACKET TYPE
01756P 05DF B1 02    A        CMPA   #2       COMPARE WITH A SARM COMMAND
01757P 05E1 27 91 0574        BEQ    RSARM    RECEIVED AN SARM COMMAND IF VALUE IS ZERO

01759P 05E3 B1 03    A        CMPA   #3       COMPARE WITH A DISCONNECT COMMAND
01760P 05E5 27 BD 05A4        BEQ    ISSDSC   RECEIVED A DISCONNECT PACKET COMMAND
01761P 05E7 6F 00    A        CLR    0,X      DEVALIDATE BUFFER
01762P 05E9 39                RTS
```

```
01764                    *  WAIT FOR BUFFER

01766                    *  THIS ROUTINE WILL WAIT FOR  THE (INCOMD) BUFFER TO
01767                    *  BECOME VALID.  IT WILL CYCLE ENDLESSLY UNTIL A COMMAND IS
01768                    *  RECEIVED.  WHILE IT WAITS, IT WILL PASS CONTROL BACK TO
01769                    *  THE SCHEDULER PER CYCLE.  ONCE THE BUFFER IS VALID,
01770                    *  THE PRIMARY AND SECONDARY POINTERS ARE SWAPPED.



01772           05EA  P RWAIT  EQU    *
01773P 05EA FE 1764  D         LDX    STDBY5   FETCH THE SECONARY POINTER
01774P 05ED 6F 00    A         CLR    0,X      DEVALIDATE IT
01775P 05EF FE 1762  D RWAIT2  LDX    INCOMD   FETCH THE PRIMARY POINTER
01776P 05F2 A6 00    A         LDAA   0,X      INTERROGATE THE STATUS WORD
01777P 05F4 26 03 05F9         BNE    *+5      BUFFER NOT VALID IF VALUE IS ZERO
01778P 05F6 3F                 SWI             PASS CONTROL TO SCHEDULER
01779P 05F7 20 F6 05EF         BRA    RWAIT2   REPEAT LOOP


01781                    *  BUFFER IS VALID, EXCHANGE PRIMARY AND SECONDARY POINTERS.

01783P 05F9 0F                 SEI             ENTERING CRITICAL REGION
01784P 05FA 7C 0000  A         INC    CRTFLG
01785P 05FD FF 1774  D         STX    TEMPR5
01786P 0600 FE 1764  D         LDX    STDBY5
01787P 0603 FF 1762  D         STX    INCOMD
01788P 0606 FE 1774  D         LDX    TEMPR5
01789P 0609 FF 1764  D         STX    STDBY5
01790P 060C 7A 0000  A         DEC    CRTFLG   EXITING CRITICAL REGION
01791P 060F 0E                 CLI
01792P 0610 39                 RTS
```

```
01794                    *  ISSUE RR RESPONSE

01796                    *  THIS ROUTINE WILL ISSUE AN RR RESPONSE AND CLEAR THE VARIABLES
01797                    *  THAT PROHIBIT A NORMAL CONOITION.  THIS ROUTINE WILL WAIT
01798                    *  FOR THE RESPONSE BUFFER TO BE EMPTY.

01800           0611  P ISSRR  EQU    *
01801P 0611 CE 175E  D         LDX    #OUTRES  FETCH THE PRIMARY POINTER
01802P 0614 BD 00BD  P         JSR    WAITBF   WAIT FOR THE BUFFER TO BECOME EMPTY
01803P 0617 5F                 CLRB            CLEAR THE FOLLOWING VARIABLES:
01804P 0618 F7 178A  D         STAB   HBUSY    HOST BUSY FLAG
01805P 061B F7 178B  D         STAB   HREJ     HOST REJECTION FLAG
01806P 061E F7 1799  0         STAB   FINBIT   RESPONOER FINAL BIT
01807P 0621 C6 04    A         LOAB   #4       LOAD RR COOE
0180BP 0623 7E 065B  P         JMP    RLOAD    ISSUE RESPONSE
```

```
01810                   *  ISSUE REJECTION RESPONSE

01812                   *  IN ISSUING A REJECTION RESPONSE, SEVERAL TASKS MUST BE PERFORMED.
01813                   *  IF THE HOST REJECTIION FLAG IS SET, THE PACKET IS IGNORED.  THE
01814                   *  PACKET IS ALSO IGNORED IF THE HOST BUSY FLAG IS SET.  FINALLY,
01815                   *  A REJECTION RESPONSE, IS ISSUED AND THE PACKET IS IGNORED.



01817           0626  P ISSREJ EQU    *
01818P 0626 B6 1788   0         LDAA   HREJ     INTERROGATE THE HOST REJECTION FLAG
01819P 0629 26 1A 0645          BNE    RJEXT1   FLAG IS CLEAR IF VALUE IS ZERO
01820P 0628 B6 01     A         LDAA   #1
01821P 0620 B7 1788   0         STAA   HREJ     SET HOST REJECTION FLAG

01823P 0630 B6 178A   0         LDAA   HBUSY    INTERROGATE HOST BUSY FLAG
01824P 0633 26 10 0645          BNE    RJEXT1   FLAG IS CLEAR IF VALUE IS ZERO

01826P 0635 B6 17A2   0         LDAA   HONLNR   INTERROGATE THE HOST ON LINE RECEIVER FLAG
01827P 0638 27 BC 05C6          BEQ    ISCMOR   OFF LINE IF BRANCH IS TAKEN

01829P 063A CE 175E   0         LDX    #OUTRES  FETCH THE PRIMARY POINTER
01830P 0630 BD 00B0   P         JSR    WAITBF   WAIT FOR THE BUFFER TO BECOME EMPTY
01831P 0640 C6 06     A         LDAB   #6       LOAD WITH REJECTION CODE
01832P 0642 BD 0658   P         JSR    RLOAD    ISSUE REJECTION RESPONSE
01833P 0645 FE 1764   0 RJEXT1  LDX    STOBY5   FETCH SECONDARY BUFFER OF (INCOMO)
01834P 0648 7E 0550   P         JMP    HEXIT    IGNORE RECEIVED PACKET



01836                   *  CLEAR BUSY

01838                   *  THIS SMALL ROUTINE WILL CLEAR THE HOST BUSY FLAG AND SET
01839                   *  THE HOST BUSY INTERMEDIATE FLAG.  THIS USUALLY OCCURS WHEN
01840                   *  THE LINK IS BEGINNING RESTART PROCEDURES AND THE INITIATOR
01841                   *  HAS NOT TRANSMITTED THE CORRECT SEQUENCE NUMBER.



01843           0648  P CLRBSY EQU    *
01844P 0648 4F                  CLRA            CLEAR THE FOLLOWING:
01845P 064C B7 178A   0         STAA   HBUSY    HOST BUSY FLAG
01846P 064F B7 1793   0         STAA   RCOUNT   REPEAT COUNTER

01848P 0652 7C 1796   0         INC    HCBUSY   SET THE HOST INTERMEDIATE FLAG
01849P 0655 7E 0611   P         JMP    ISSRR    ISSUE AN RR RESPONSE
```

```
01851                    *  ISSUE A RESPONSE

01853                    *  THIS ROUTINE WILL ISSUE A RESPONSE PACKET.  THE COOE FOR THE
01854                    *  RESPONSE IS PASSEO IN ACCUMULATOR 8.  ACCUMULATOR A IS THE
01855                    *  WORKING REGISTER ANO IS UNOEFINEO WHEN EXITING.



01857           0658  P RLOAO  EQU    *
01858P 0658 86 1780   O        LOAA   VOFR     FETCH THE ACKNOWLEOGING SEQUENCE NUM8ER
01859P 0658 A7 05     A        STAA   5,X      INITIALIZE THE 8UFFER WITH THE SEQUENCE NUM8ER

01861P 065O 86 1799   O        LOAA   FIN8IT   FETCH THE FINAL 8IT
01862P 066O A7 03     A        STAA   3,X      INITIALIZE THE 8UFFER WITH THE FINAL 8IT
01863P 0662 E7 01     A        STA8   1,X      INITIALIZE THE 8UFFER WITH THE PACKET TYPE
01864P 0664 E7 00     A        STA8   0,X      VALIOATE THE 8UFFER

01866P 0666 FE 1764   O        LOX    STO8Y5   FETCH THE SECONOARY AOORESS FOR (INCOMO)
01867P 0669 6F 00     A        CLR    0,X      OEVALIOATE THE COMMANO 8UFFER
01868P 0668 39          REJEXT RTS



01870                    *  ISSUE RNR RESPONSE

01872                    *  THE RNR RESPONSE IS VARY SIMULAR TO THE RR RESPONSE.  THE ONLY
01873                    *  OIFFERENCE IS THAT THE RNR RESPONSE OECLEARES THE HOST
01874                    *  TO 8E 8USY.  OTHEREWISE, THEY ARE IOENTICAL.


01876           066C  P ISSRNR EQU    *
01877P 066C CE 175E   O        LOX    #OUTRES  FETCH THE PRIMARY POINTER
01878P 066F 8O 0080   P        JSR    WAIT8F   WAIT FOR 8UFFER TO 8ECOME EMPTY
01879P 0672 86 01     A        LOAA   #1
01880P 0674 87 178A   O        STAA   H8USY    SET HOST 8USY FLAG TO ONE
01881P 0677 7F 1799   O        CLR    FIN8IT   CLEAR FINAL 8IT OF RESPONOER
01882P 067A 7F 1788   O        CLR    HREJ     CLEAR HOST REJECTION FLAG
01883P 067O C6 05     A        LOA8   #5       LOAO THE COOE FOR A RNR RESPONSE
01884P 067F 7E 0658   P        JMP    RLOAO    ISSUE RESPONSE
```

```
01886                  ************************************************************************
01887                  *                                                                      *
01888                  *    TRANSMITTER                                                       *
01889                  *                                                                      *
01890                  *    THE TRANSMITTER RECEIVES COMMANDS FROM THE INITIATOR AND          *
01891                  *    RESPONSES FROM THE RESPONDER.  THE PURPOSE OF THE TRANSMITTER     *
01892                  *    IS TO FORM EITHER COMMAND PACKETS OR RESPONSE PACKETS             *
01893                  *    AND TRANSMIT THEM AS A WHOLE TO THE TRANSMITTING PORT.            *
01894                  *    THIS ROUTINE CALCULATES THE CHECKSUM BY ADDING ALL THE BYTES      *
01895                  *    WHICH ARE SENT IN THE PACKET.  THE TRANSMITTER ALSO USES          *
01896                  *    TRANSPARENCY BYTES TO ELIMINATE CONFUSION BETWEEN A               *
01897                  *    CONTROL PHASE AND A DATA BYTE.  ALL COMMANDS AND RESPONSES ARE    *
01898                  *    TRANSMITTED AS RECEIVED EXCEPT AN INFORMATION PACKET AND A        *
01899                  *    RR PACKET.  AN RR PACKET MAY RIDE PIGGY-BACK ON THE INFORMATION   *
01900                  *    PACKET, THUS, ELIMINATING ONE PACKET.  EACH PACKET MAY            *
01901                  *    HAVE UP TO 128 BYTES IN IT.  THE NORMAL PROCESS LOOP WILL POLL    *
01902                  *    THE TWO INPUT SECONDARY BUFFERS FOR A JOB TO PERFORM.  AT         *
01903                  *    THE END OF THE CYCLIC LOOP, CONTROL IS RETURNED TO THE            *
01904                  *    SCHEDULER.  WHEN THIS ROUTINE RECEIVES CONTROL ONCE AGAIN,        *
01905                  *    THE LOOP IS REPEATED.                                             *
01906                  *                                                                      *
01907                  ************************************************************************



01909          0000  A        IFEQ   CHANGE   ASSEMBLED FOR (DTE) IF THE VALUE IS ZERO
01910          0682  P TKPC2  EQU    *
01911                         ENDC

01913          0000  A        IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
01914                  TKPC9 EQU *
01915                    ENDC


01917                  *  INTERROGATE THE COMMAND BUFFER

01919          0682  P XMITER EQU    *
01920P 0682 FE 1756  D TRYCMD LDX    OUTCMD   FETCH THE PRIMARY BUFFER POINTER
01921P 0685 A6 00    A        LDAA   0,X      INTERROGATE THE STATUS WORD
01922P 0687 27 1A 06A3        BEQ    TRYRES   BUFFER IS EMPTY IF VALUE IS ZERO
01923P 0689 0F                SEI             ENTERING CRITICAL REGION
01924P 068A 7C 0000  A        INC    CRTFLG
01925P 068D FF 1770  D        STX    TEMPR3
01926P 0690 FE 1758  D        LDX    STOBY2
01927P 0693 FF 1756  D        STX    OUTCMD
01928P 0696 FE 1770  D        LDX    TEMPR3
01929P 0699 FF 1758  D        STX    STOBY2
01930P 069C 7A 0000  A        DEC    CRTFLG   EXITING CRITICAL REGION
01931P 069F 0E                CLI
01932P 06A0 BD 06EE  P        JSR    COMMND   VALID COMMAND, TRANSMIT IT
```

```
01934                   *  CHECK FOR POSSIBLE RR FOR PIGGY-BACK

01936P 06A3 70 179B  D TRYRES TST    RRFLG    INTERROGATE RECEIVE READY FLAG
01937P 06A6 27 03 06AB        BEQ    *+5      FLAG SET IF VALUE IS NON-ZERO
0193BP 06AB BD 06DF  P        JSR    RESXMT   TRANSMIT RR ON PIGGY-BACK


01940                   *  POLL THE RESPONSE BUFFER.

01942P 06AB FE 175E  D        LDX    OUTRES   FETCH THE PRIMARY POINTER
01943P 06AE A6 00    A        LDAA   0,X      INTERROGATE THE STATUS WORD
01944P 06B0 27 1A 06CC        BEQ    XTWAIT   BUFFER IS EMPTY IF VALUE IS ZERO

01946P 06B2 0F                SEI             ENTERING CRITICAL REGION
01947P 06B3 7C 0000  A        INC    CRTFLG
0194BP 06B6 FF 1770  D        STX    TEMPR3
01949P 06B9 FE 1760  D        LDX    STDBY4
0195OP 06BC FF 175E  D        STX    OUTRES
01951P 06BF FE 1770  D        LDX    TEMPR3
01952P 06C2 FF 1760  D        STX    STDBY4
01953P 06C5 7A 0000  A        DEC    CRTFLG   EXITING CRITICAL REGION
01954P 06CB 0E                CLI
01955P 06C9 BD 06CF  P        JSR    RESPNC   TRANSMIT RESPONSE

01957P 06CC 3F         XTWAIT SWI             RETURN CONTROL TO THE SCHEDULER
0195BP 06CD 20 B3 0682        BRA    TRYCMD   REPEAT THE LOOP
```

```
01960                    *  TRANSMIT A RESPONSE

01962                    *  THIS ROUTINE WILL TRANSMIT A RESPONSE.  IF THE RESPONSE IS A RR,
01963                    *  (RECEIVE READY) A FLAG WILL BE SET (RRFLG).  THIS WILL SIGNAL
01964                    *  THE TRANSMITTER THAT A POSSIBLE PIGGY-BACK OPERATION COULD
01965                    *  TAKE PLACE.  FOR ANY OTHERE RESPONSE, THE ROUTINE (SENDRR)
01966                    *  IS CALL TO ISSUE THE PACKET.



0196B          06CF  P RESPNC EQU    *
01969P 06CF 20 0E 06DF        BRA    RESXMT   BYPASS PIGGY-BACK OPERATION
0197OP 06D1 A6 01     A       LDAA   1,X      FETCH PACKET TYPE
01971P 06D3 B1 04     A       CMPA   #4       COMPARE PACKET TYPE WITH A RR CODE
01972P 06D5 26 0B 06DF        BNE    RESXMT   RR RESPONSE IF VALUE IS ZERO

01974P 06D7 A6 03     A       LDAA   3,X      FETCH FINAL BIT
01975P 06D9 26 04 06DF        BNE    RESXMT   BIT IS ZERO IF VALUE ZERO

01977P 06DB 7C 179B   D       INC    RRFLG    SET FLAG FOR PIGGY-BACK OPTION
0197BP 06DE 39                RTS

01980                    *  TRANSMIT A RESPONSE

019B2P 06DF FE 1760   D RESXMT LDX    STDBY4   FETCH SECONDARY BUFFER POINTER
019B3P 06E2 A6 05     A       LDAA   5,X      FETCH ACKNOWLEDGING SEQUENCE NUMBER
019B4P 06E4 B7 1797   D       STAA   NOFR     SAVE FOR TRANSMITTER RETRIEVAL
019B5P 06E7 7F 179B   D       CLR    RRFLG    CLEAR PIGGY-BACK OPTION FLAG
019B6P 06EA BD 071C   P       JSR    SENDRR   ISSUE RESPONSE
019B7P 06ED 39                RTS
```

```
019B9                    *  TRANSMIT A COMMAND

01991                    *  THIS ROUTINE WILL TRANSMIT A COMMAND PACKET.  IF THE COMMAND
01992                    *  IS AN INFORMATION PACKET, THE (RRFLG) IS SENSED.  IF
01993                    *  IT IS SET, A PIGGY-BACK OPERATION WILL TAKE PLACE.



01995           06EE   P COMMND EQU    *
01996P 06EE A6 01      A        LDAA   1,X       FETCH THE PACKET TYPE
01997P 06F0 B1 01      A        CMPA   #1        COMPARE WITH AN INFORMATION PACKET
0199BP 06F2 26 16 070A          BNE    NOIFRM    INFORMATION PACKET IF VALUE IS ZERO


02000                    *  THE PACKET IS AN INFORMATION PACKET.  EXAMINE THE (RRFLG)
02001                    *  ANO DETERMINE WHETHER A PIGGY-BACK OPERATION IS POSSIBLE.

02003P 06F4 7D 179B  D          TST    RRFLG     INTERROGATE THE PIGGY-BACK FLAG
02004P 06F7 27 0D 0706          BEQ    XMTSHT    PIGGY-BACK NOT POSSIBLE IF VALUE ZERO


02006                    *  PIGGY-BACK IS POSSIBLE.  RETRIEVE THE ACKNOWLEDGING
02007                    *  SEQUENCE NUMBER, DEVALIDATE RR BUFFER, AND TRANSMIT.

02009P 06F9 FE 1760  D          LDX    STDBY4    FETCH SECONARY BUFFER FOR (OUTRES)
0200AP 06FC A6 05      A        LDAA   5,X       FETCH ACKNOWLEDGING SEQUENCE NUMBER
0200BP 06FE B7 1797  D          STAA   NOFR      UPDATE ACKNOWLEDGING POINTER
0200CP 0701 6F 00      A        CLR    0,X       DEVALIDATE BUFFER
0200DP 0703 7F 179B  D          CLR    RRFLG     CLEAR PIGGY-BACK FLAG

0200FP 0706 BD 072D  P XMTSHT JSR    XMTCMD    TRANSMIT COMMAND
02010P 0709 39                  RTS
```

```
02018                   *  NO I-FRAME

02020                   *  THIS ROUTINE WILL TRANSMIT A COMMAND PACKET OTHER THAN AN
02021                   *  INFORMATION PACKET.  SINCE THE PIGGY-BACK OPTION IS NOT POSSIBLE
02022                   *  HERE, AN RR PACKET WILL BE ISSUED IF THE (RRFLG) IS SET.
02023                   *  THE COMMAND PACKET WILL FOLLOW IT.


02025P 070A 70 179B  0 NOIFRM TST    RRFLG    TEST THE PIGGY-BACK FLAG
02026P 0700 27 F7 0706        BEQ    XMTSHT   NO RESPONSE IF VALUE IS ZERO



02028                   *  THERE IS A RESPONSE PENDING.  TRANSMIT
02029                   *  THE RESPONSE FOLLOWED BY THE COMMAND PACKET.

02031P 070F 7F 179B  0        CLR    RRFLG    CLEAR PIGGY-BACK FLAG
02032P 0712 FE 1760  0        LDX    STDBY4   FETCH SECONDARY BUFFER DF (DUTRES)
02033P 0715 BD 071C  P        JSR    SENDRR   TRANSMIT RESPONSE
02034P 0718 BD 0706  P        JSR    XMTSHT   TRANSMIT THE COMMAND PACKET
02035P 071B 39                RTS



02037                   *  ISSUE A RESPONSE

02039                   *  THIS ROUTINE WILL INITIALIZE THE CORRECT CHANNEL, BUILD
02040                   *  THE CONTROL WORD, AND TRANSMIT THE PACKET.  THE RESPONSE
02041                   *  BUFFER IS THEN CLEARED.


02043          071C  P SENDRR EQU    *
02044          0000  A        IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
02045P 071C B6 03    A        LDAA   #3       CHANNEL DEFINITION FOR RESPONSE DF (DTE)
02046                         ENDC

02048          0000  A        IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
02049                     LDA A #1  CHANNEL DEFINITION FOR RESPONDER DF (DCE)
02050                     ENDC
02051P 071E B7 179E  0        STAA   CHANNL   INITIAL THE CHANNEL VARIABLE

02053P 0721 BD 0740  P        JSR    CNTLWD   CONSTRUCT CONTROL WORD
02054P 0724 BD 07AD  P        JSR    XMTFRM   TRANSMIT FRAME

02056P 0727 FE 1760  0        LDX    STDBY4   FETCH THE SECONDARY BUFFER DF (DUTRES)
02057P 072A 6F 00    A        CLR    0,X      DEVALIDATE THE BUFFER
02058P 072C 39                RTS
```

```
02060                   *  TRANSMIT COMMAND

02062                   *  THIS ROUTINE WILL TRANSMIT A COMMAND PACKET.  THIS RDUTINE
02063                   *  WILL INITIALIZE THE BUFFER, DETERMINE THE CORRECT
02064                   *  CHANNEL, AND TRANSMIT IT.  THE NECESSARY INFORMATION IS THE
02065                   *  NODE ADDRESS, THE FINAL BIT, THE BYTE COUNT, AND PACKET TYPE.



02067           072D  P XMTCMD EQU    *
02068P 072D FE 175B  D         LDX    STDBY2   FETCH SECONDARY POINTER OF (OUTCMD)
02069P 0730 BD 074D  P         JSR    CNTLWD   BUILD CONTROL WORD

02071P 0733 A6 07    A         LDAA   7,X      FETCH LSB OF NODE ADDRESS
02072P 0735 B7 177B  D         STAA   XMTMP+1  INITIALIZE LSB OF FRAME ADDRESS
02073P 0738 A6 06    A         LDAA   6,X      FETCH MSB OF NODE ADDRESS
02074P 073A B7 177A  D         STAA   XMTMP    INITIALIZE MSB OF FRAME ADDRESS
02075P 073D A6 02    A         LDAA   2,X      FETCH THE BYTE COUNT
02076P 073F B7 17A0  D         STAA   XTBYTE   INITIALIZE FRAME COUNTER
02077P 0742 6F 00    A         CLR    0,X      DEVALIDATE BUFFER

02079           0000  A         IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
02080P 0744 B6 01    A         LDAA   #1       COMMAND CHANNEL FOR (DTE)
02081                          ENDC
02082           0000  A         IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
02083                     LDA A #3  COMMAND CHANNEL FOR (DCE)
02084                     ENDC
02085P 0746 B7 179E  D         STAA   CHANNL   INITIALIZE CHANNEL VARIABLE FOR FRAME TRANSMITTER
02086P 0749 BD 07AD  P         JSR    XMTFRM   TRANSMIT FRAME
02087P 074C 39                RTS
```

```
02089                    *  BUILD CONTROL WORD

02091                    *  THE CONTROL WORD IS THE SECOND BYTE WHICH IS TRANSMITTED
02092                    *  IN THE FRAME.  IT CONTAINS THE NECESSARY INFORMATION ON THE
02093                    *  PACKET TYPE, SEQUENCE NUMBER, AND ACKNOWLEDGEMENT NUMBER, IF ANY.
02094                    *  THE ROUTINE BUILDS THE CONTROL BY FIRST FINDING THE BASE WORD.
02095                    *  ONTO THE BASE WORD, THE POLL/FINAL BIT IS SUPERIMPOSED.  THE
02096                    *  SEQUENCE NUMBER IS ADDED AND SO IS THE ACKNOWLEDGEMENT NUMBER
02097                    *  IF APPLICABLE TO THE PACKET.



02099           074D  P CNTLWD EQU    *
02100P 074D A6 01     A         LDAA   1,X       FETCH THE PACKET TYPE
02101P 074F B7 17A4   D         STAA   XMTYPE    SAVE FOR FUTURE USE
02102P 0752 16                  TAB
02103P 0753 BD 076D   P         JSR    CNTRWD    FIND BASE WORD
02104P 0756 BD 0797   P         JSR    BLDCTR    SUPERIMPOSE THE ACKNOWLEDGING NUMBER (NOFR)

02106P 0759 6D 03     A         TST    3,X       INTERROGATE FINAL BIT
02107P 075B 27 02 075F          BEQ    *+4       FINAL BIT IS ZERO IF VALUE IS ZERO
02108P 075D BA 10     A         ORAA   #$10     SUPERIMPOSE FINAL BIT

02110P 075F E6 01     A         LDAB   1,X       FETCH PACKET TYPE
02111P 0761 C1 01     A         CMPB   #1        COMPARE FOR AN IFORMATION PACKET
02112P 0763 26 04 0769          BNE    *+6       IF INFORMATION PACKET VALUE IS ZERO


02114                    *  ADD SEQUENCE NUMBER ONLY IF THIS IS AN I-FRAME PACKET.

02116P 0765 E6 04     A         LDAB   4,X       FETCH THE SEQUENCE NUMBER
02117P 0767 5B                  ASLB
02118P 0768 1B                  ABA              SUPERIMPOSE INTO CONTROL WORD

02120P 0769 B7 17A3   D         STAA   XTCNTL    STORE CONTROL WORD FOR LATER RETRIEVAL
02121P 076C 39                  RTS
```

```
02123                    *  FIND 8ASE WORD

02125                    *  THIS ROUTINE WILL RETURN THE BASE WORD OF THE FINAL CONTROL WORD
02126                    *  TO ITS CALLER.  THE PACKET TYPE IS PASSED TO THIS ROUTINE IN
02127                    *  ACCUMULATOR A.  THE RETURNED BASE VALUE IS ALSO CONTAINED IN
02128                    *  ACCUMULATOR A.  ACCUMULATOR 8 IS UNEFFECTED.



02130           076D  P CNTRWD EQU    *
02131P 076D 81 04    A         CMPA   #4        COMPARE FOR A RR RESPONSE
02132P 076F 26 03 0774         8NE    *+5       TEST FAILS IF VALUE IS NON-ZERO
02133P 0771 86 01    A         LDAA   #1        8ASE VALUE FOR RR RESPONSE
02134P 0773 39                 RTS


02136P 0774 81 05    A         CMPA   #5        COMPARE FOR A RNR RESPONSE
02137P 0776 26 01 0779         8NE    *+3       TEST FAILS IF VALUE IS NON-ZERO
02138P 0778 39                 RTS


02140P 0779 81 06    A         CMPA   #6        COMPARE FOR A REJ RESPONSE
02141P 0778 26 03 0780         8NE    *+5       TEST FAILS IF VALUE IS NON-ZERO
02142P 077D 86 09    A         LDAA   #9        LOAD 8ASE WORD FOR REJ RESPONSE
02143P 077F 39                 RTS


02145P 0780 81 07    A         CMPA   #7        COMPARE FOR A UA RESPONSE
02146P 0782 26 03 0787         8NE    *+5       TEST FAILS IF VALUE IS NON-ZERO
02147P 0784 86 63    A         LDAA   #$63      LOAD 8ASE WORD FOR UA RESPONSE
02148P 0786 39                 RTS


0215DP 0787 81 02    A         CMPA   #2        COMPARE FOR A SARM COMMAND
02151P 0789 26 03 078E         BNE    *+5       TEST FAILS IF VALUE IS NON-ZERO
02152P 0788 86 0F    A         LDAA   #$0F      LOAD WITH 8ASE WORD FOR SARM COMMAND
02153P 078D 39                 RTS


02155P 078E 81 03    A         CMPA   #3        COMPARE FOR A DISC COMMAND
02156P 0790 26 03 0795         8NE    *+5       TEST FAILS IF VALUE IS NON-ZERO
02157P 0792 86 43    A         LDAA   #$43      LOAD WITH 8ASE WORD FOR DISC COMMAND
02158P 0794 39                 RTS


02160                    *  IF ALL TESTS FAIL, THE PACKET TYPE IS
02161                    *  AN INFORMATION PACKET, AND THE BASE
02162                    *  CONTROL WORD IS ZERO.

02164P 0795 4F                 CLRA             8ASE CONTROL WORD FOR INFORMATION PACKET
02165P 0796 39                 RTS
```

```
02167                    *  SUPERIMPOSE (NOFR)

02169                    *  THIS ROUTINE WHEN CALLED IN CONJUNCTION WITH THE OTHER ROUTINES
02170                    *  NECESSARY TO BUILD A CONTROL WORD WILL SUPERIMPOSE THE (NOFR)
02171                    *  VARIABLE INTO THE CONTROL WORD.  THE ROUTINE WILL BYPASS THE
02172                    *  SUPERPOSITION SECTION IF THE CONTROL WORD THAT IS BEING BUILT
02173                    *  IS AN UNNUMBERED COMMAND OR RESPONSE.


02175           0797  P BLOCTR EQU    *
02176P 0797 C1 01     A         CMPB   #1        COMPARE FOR A I-FRAME PACKET
02177P 0799 27 08 07A3          BEQ    BLO       I-FRAME EXISTS IF VALUE IS ZERO
02178P 079B C1 06     A         CMPB   #6        COMPARE FOR LOWER LIMIT
02179P 079D 2E 0D 07AC          BGT    BEXIT     SKIP IF PACKET CODE IS GREATER THAN 6
02180P 079F C1 04     A         CMPB   #4        COMPARE WITH UPPER LIMIT
02181P 07A1 2D 09 07AC          BLT    BEXIT     SKIP IF PACKET CODE IS LESS THAN 4

02183P 07A3 F6 1797   D BLO     LOAB   NOFR      FETCH ACKNOWLEDGING NUMBER
02184P 07A6 58                  ASLB             ADJUST FOR SUPERPOSITION
02185P 07A7 58                  ASLB
02186P 07A8 58                  ASLB
02187P 07A9 58                  ASLB
02188P 07AA 58                  ASLB
02189P 07AB 1B                  ABA              SUPERIMPOSE ACKNOWLEDGING WORD
02190P 07AC 39          BEXIT   RTS
```

```
02192                    *  TRANSMIT A FRAME

02194                    *  EVERY PACKET THAT IS SENT TO THE RESPONDER MUST BE FRAMED.
02195                    *  FRAMING ADDES THE CHANNEL NUMBER AND CONTROL BYTE TO THE
02196                    *  ALREADY EXITING PACKET.  AT THE END, A CHECKSUM IS SENT, TOO.
02197                    *  TRANSPARENCY BYTES ARE INSERTED TO PREVENT CONFUSION
02198                    *  WITH CONTROL BYTES.



02200            07AD  P XMTFRM EQU    *
02201P 07AD 4F                  CLRA            CLEAR THE FOLLOWING:
02202P 07AE B7 17B5  D          STAA   XTCHCK+1 LSB OF TRASNMITTING CHECKSUM
02203P 07B1 B7 17B4  D          STAA   XTCHCK   MSB OF TRANSMITTING CHECKSUM

02205            0000  A        IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
02206P 07B4 B6 00     A        LDAA   #NETWRK  FETCH THE PORT NUMBER TOWARD COMMUNICATION LINK
02207                          ENDC

02209            0000  A        IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
02210                       LDA A #SUBNET  LOAD WITH PORT NUMBER OF COMMUNICATIONS LINK
02211                       ENDC

02213P 07B6 C6 10     A          LDAB   #$10     LOAD WITH ASCII (DLE)
02214P 07B8 BD 0000   A          JSR    PUTCHR   ISSUE CONTROL COMMAND
02215P 07BB BD 0B35   P          JSR    XMTSUM   ADD TO CHECKSUM

02217P 07BE C6 02     A          LDAB   #$02     LOAD WITH ASCII (STX)
02218P 07C0 BD 0000   A          JSR    PUTCHR   ISSUE ASCII START OF TEXT CODE
02219P 07C3 BD 0B35   P          JSR    XMTSUM   ADD TO CHECKSUM

02221P 07C6 F6 179E   D          LDAB   CHANNL   FETCH APPROPRIATE CHANNEL
02222P 07C9 BD 0000   A          JSR    PUTCHR   ISSUE CHANNEL NUMBER
02223P 07CC BD 0B35   P          JSR    XMTSUM   ADD TO CHECKSUM

02225P 07CF F6 17A3   D          LDAB   XTCNTL   FETCH CONTROL WORD
02226P 07D2 BD 0000   A          JSR    PUTCHR   ISSUE CONTROL WORD
02227P 07D5 BD 0B35   P          JSR    XMTSUM   AOD TO CHECKSUM
02228P 07D8 C1 10     A          CMPB   #$10     COMPARE FOR TRANSPARENCY
02229P 07DA 26 06 07E2           BNE    *+8      TRANSPARENCY NEEDED IF VALUE IS ZERO

02231P 07DC BD 0000   A          JSR    PUTCHR   ISSUE TRANSPARENCY BYTE
02232P 07DF BD 0B35   P          JSR    XMTSUM   ADD TO CHECKSUM
```

```
02234                    *  THE PACKET THAT IS BEING FRAMED IS INTERROGATED AS A
02235                    *  POSSIBLE I-FRAME.  IF IT IS, THEN THE TRANSMITTING DATA
02236                    *  ROUTINE IS CALLED TO ISSUE THE DATA.  ELSE, THE ROUTINE
02237                    *  IS SKIPPED.  THERE IS NO OTHER PACKET WHICH CONTAINS DATA.



02239P 07E2 F6 17A4  D        LDAB   XMTYPE   FETCH PACKET TYPE
0224OP 07E5 C1 01    A        CMPB   #1       COMPARE FOR AN I-FRAME
02241P 07E7 26 03 07EC        BNE    *+5      AN I-FRAME EXISTS IF THE VALUE IS ZERO
02242P 07E9 BD 0B13  P        JSR    XMTDAT   TRANSMIT DATA
02243P 07EC C6 10    A        LDAB   #$10     LOAD WITH ASCII (DLE)
02244P 07EE BD 0000  A        JSR    PUTCHR   ISSUE (DLE) CODE
02245P 07F1 BD 0B35  P        JSR    XMTSUM   ADD TO CHECKSUM

02247P 07F4 C6 03    A        LDAB   #$03     LOAD WITH ASCII (ETX)
0224BP 07F6 BD 0000  A        JSR    PUTCHR   ISSUE COMMAND WORD
02249P 07F9 BD 0B35  P        JSR    XMTSUM   ADD TO CHECKSUM


02251                    *  CALCULATE CHECKSUM.  TAKE THE
02252                    *  NEGATION OF IT FOR TRANSMISSION PURPOSES.

02254P 07FC B6 17B5  D        LDAA   XTCHCK+1 FETCH THE LSB OF CHECKSUM
02255P 07FF F6 17B4  D        LDAB   XTCHCK   FETCH THE MSB OF CHECKSUM
02256P 0B02 43                COMA           COMPLEMENT CHECKSUM
02257P 0B03 53                COMB
0225BP 0B04 BB 01    A        ADDA   #1       TAKE TWO COMPLEMENT
02259P 0B06 C9 00    A        ADCB   #0
0226OP 0B0B 36                PSHA

02262          0000  A        IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
02263P 0B09 B6 00    A        LDAA   #NETWRK  FETCH PORT NUMBER
02264                         ENDC

02266          0000  A        IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
02267                    LDA A #SUBNET  FETCH PORT NUMBER
0226B                    ENDC

0227OP 0B0B BD 0000  A        JSR    PUTCHR   ISSUE MSB TO PORT
02271P 0B0E 33                PULB
02272P 0B0F BD 0000  A        JSR    PUTCHR   ISSUE LSB TO PORT
02273P 0B12 39                 RTS
```

```
D2275                    *  TRANSMIT DATA

D2277                    *  THE DNLY PACKET WHICH CDNTAINS DATA IS THE INFDRMATIDN
D2278                    *  PACKET.  ALL DTHER PACKET TRANSMISSIDNS WILL SKIP DVER THIS
D2279                    *  RDUTINE.  THERE IS A MAXIMUM DF 128 8YTES PER DATA PACKET
D228D                    *  WHICH DDES NDT INCLUDE THE PDSSI8ILITY DF TRANSPARENCY BYTES.
D2281                    *  TRANSPARENCY 8YTES ARE ISSUED IF THE DATA WDRD 8EING TRANSMITTED
D2282                    *  HAS A 8IT PATTERN EQUIVALENT TD AN ASCII (DLE), A [HEX 1D].
D2283                    *  A CHECKSUM IS MAINTAINED.



D2285          D813  P XMTDAT EQU    *
D2286P D813 FE 177A  D        LDX    XMTMP     FETCH DATA ADDRESS NDDE
D2287P D816 FF 177D  D        STX    TEMPR3    SAVE IN WDRKING REGISTER
D2288P D819 FE 177D  D XMTLDP LDX    TEMPR3    FETCH ADRESS DF NEXT DATA WDRD
D2289P D81C E6 DD    A        LDA8   D,X       FETCH DATA WDRD
D229DP D81E D8                INX              INCREASE ADDRESS PDINTER 8Y DNE 8YTE
D2291P D81F FF 177D  D        STX    TEMPR3    UPDATA WDRKING REGISTER
D2292P D822 8D DDDD  A        JSR    PUTCHR    RETURN NDDE TD THE NDDE PDDL
D2293P D825 8D D835  P        JSR    XMTSUM    ADD THE DATA 8YTE INTD THE CHECKSUM

D2295P D828 C1 1D    A        CMP8   #$1D      CDMPARE FDR TRANSPARENCY DPTIDN
D2296P D82A 26 D3 D82F        8NE    *+5       TRANSPARENCY DPTIDN NEEDED IF VALUE IS ZERD
D2297P D82C 8D DDDD  A        JSR    PUTCHR    ISSUE TRANSPARENCY 8YTE
D2298P D82F 7A 17AD  D        DEC    XT8YTE    DECREMENT 8YTE CDUNT
D2299P D832 26 E5 D819        8NE    XMTLDP    REPEAT TRANSMISSIDN UNTIL 8YTE CDUNT REACHES ZERD
D23DDP D834 39                RTS



D23D2                    *  TRASNSMITTING CHECKSUM

D23D4                    *  A CHECKSUM IS MAINTAINED THRDUGHDUT THE TRANSMISSIDN DF THE
D23D5                    *  PACKET.  THE DATA 8YTE WHICH HAS A LENGTH DF EIGHT 8ITS IS
D23D6                    *  ADDED INTD THE CHECKSUM REGISTER.  A CDNTINUING SUM IS
D23D7                    *  MAINTAINED IN THE REGISTER AND IT IS SIXTEEN 8ITS IN LENGTH.
D23D8                    *  THE ADDITIDN IS 8ASED DN MDDULD 65,536.



D231D          D835  P XMTSUM EQU    *
D2311P D835 37                PSH8             SAVE ACCUMULATDR 8
D2312P D836 F8 1785  D        ADD8   XTCHCK+1   ADD DATA 8YTE TD LS8 DF CHECKSUM
D2313P D839 F7 1785  D        STA8   XTCHCK+1   UPDATE LS8 DF CHECKSUM
D2314P D83C 24 D3 D841        8CC    *+5        DVERFLDW (CARRY) EXISTS IF 8RANCH IS NDT TAKEN
D2315P D83E 7C 1784  D        INC    XTCHCK     ADJUST MS8 DF CHECKSUM FDR DVERFLDW (CARRY)
D2316P D841 33                PUL8
D2317P D842 39                RTS
```

```
02319                 ************************************************************************
02320                 *                                                                      *
02321                 *    RECEIVER                                                          *
02322                 *                                                                      *
02323                 *    THE RECEIVER RECEIVES AN ENTIRE FRAME FROM ITS CORRESPONDING      *
02324                 *    TRANSMITTER AT THE OTHER END OF THE LINK.  AS THE FRAME IS        *
02325                 *    BEING RECEIVED,  THE TRANSPARENCY BYTES ARE STRIPPED OFF AND A    *
02326                 *    CHECKSUM IS MAINTAINED.  A NODE IS FETCHED FROM THE NODE POOL     *
02327                 *    BY THIS ROUTINE WHEN A FRAME RECEPTION IS SENSED.  AT THE END     *
0232B                 *    OF THE RECEPTION, THE GENERATED CHECKSUM IS COMPARED WITH          *
02329                 *    THE RECEIVED CHECKSUM.  IF THEY MATCH, THE PACKET IS CONSIDERED   *
02330                 *    VALID.  (NOTE:  A VALID CHECKSUM TEST ONLY INDICATES A FLAWLESS   *
02331                 *    TRANSMISSION.  THE PACKET ITSELF COULD BE OUT OF SYNC AND         *
02332                 *    THEREFORE, INVALID.)  THE INFORMATION ON THE PACKET IS EITHER     *
02333                 *    SENT TO THE RESPONDER (INRESP) OR THE INITIATOR (INCOMD)          *
02334                 *    DEPENDING ON THE TYPE OF PACKET.                                  *
02335                 *                                                                      *
02336                 ************************************************************************



0233B          0000  A        IFEQ   CHANGE   ASSEMBLED FOR (DTE) IF VALUE IS ZERO
02339          0843  P TKPC3  EQU    *
0234D                         ENDC

02342          0000  A        IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
02343                TKPCA EQU *
02344                  ENDC


02346P 0B43 BD 08B6  P RECEIV JSR    DECODE   RECEIVE AND DECODE PACKET

02348P 0B46 FE 177E  D        LDX    AVAILN   FETCH THE NODE ADDRESS
02349P 0849 E6 00    A        LDAB   0,X      FETCH THE CHANNEL NUMBER FROM FRAME
```

```
02351                   *  A OETERMINATION IS MAOE ON THE OESTINATION OF THE PACKET.
02352                   *  A TEST IS MAOE TO THE CHANNEL NUMBER.  IF THE COOE GENERATEO
02353                   *  IS FOR A (OTE),  THEN A CHANNEL NUMBER OF ONE WOULO IMPLY
02354                   *  A RESPONSE PACKET.  A CHANNEL NUMBER OF THREE WOULO IMPLY A
02355                   *  COMMANO.  IF THE COOE GENERATEO IS FOR A (OCE) THE REVERSE
02356                   *  IS TRUE.

0235B           0000  A         IFEQ   CHANGE   ASSEMBLE FOR (OTE) IF VALUE IS ZERO
02359P 0B4B C1 01     A         CMPB   #1        COMPARE FOR RESPONSE PACKET
02360P 0B4O 27 09 0B5B          BEQ    CHANNB   RESPONSE PACKET EXISTS IF VALUE IS ZERO
02361P 0B4F C1 03     A         CMPB   #3        COMPARE FOR A COMMANO PACKET
02362                           ENOC

02364           0000  A         IFNE   CHANGE   ASSEMBLE FOR A (OCE) IF VALUE IS NON-ZERO
02365                     CMP B #3  COMPARE FOR A RESPONSE PACKET
02366                     BEQ CHANNB  A RESPONSE PACKET EXISTS IF THE VALUE IS ZERO
02367                     CMP B #1  COMPARE FOR A COMMANO PACKET
0236B                     ENOC

02370P 0B51 26  3E 0B91         BNE    RCTURN   CHANNEL INVALIO - IGNORE IT, IF BRANCH IS TAKEN
02371P 0B53 CE 1762  0          LOX    #INCOMO  FETCH THE PRIMARY BUFFER POINTER
02372P 0B56 20 03 0B5B          BRA    *+5      SKIP AROUNO RESPONOER
02373P 0B5B CE 175A  0 CHANNB LOX    #INRESP  FETCH PRIMARY BUFFER POINTER
02374P 0B5B BO 00B0  P          JSR    WAITBF   WAIT FOR BUFFER TO BECOME EMPTY
```

```
02376                   *  THE CORRECT BUFFER HAS BEEN SELECTEO ANO IT IS EMPTY.
02377                   *  INITIALIZE THE PRIMARY BUFFER.

02379P 0B5E A7 01    A        STAA   1,X      INITIALIZE BUFFER WITH PACKET TYPE

023B1P 0B60 F6 179B  O        LOAB   POLFIN   FETCH RECEIVER POLL/FINAL BIT
023B2P 0B63 E7 03    A        STAB   3,X      INITIALIZE BUFFER WITH POLL/FINAL BIT

023B4P 0B65 F6 1790  O        LOAB   NPROFS   FETCH SEQUENCE NUMBER OF RECEIVEO PACKET
023B5P 0B6B E7 04    A        STAB   4,X      INITIALIZE BUFFER WITH RECEIVEO SEQUENCE NUMBER

023B7P 0B6A F6 17BF  O        LOAB   NT1OFR   FETCH THE ACKNOWLEOGEO PACKET SEQUENCE NUMBER
023B8P 0B6O E7 05    A        STAB   5,X      INITIALIZE BUFFER WITH ACKNOWLEOGEO SEQUENCE NUMBE

0239OP 0B6F F6 179F  O        LOAB   RCBYTE   FETCH BYTE COUNT
02391P 0B72 5A                OECB            COMPENSATE FOR FRAME OVERHEAO
02392P 0B73 5A                OECB
02393P 0B74 5A                OECB
02394P 0B75 5A                OECB
02395P 0B76 E7 02    A        STAB   2,X      INITIALIZE BUFFER WITH BYTE COUNT


02397                   *  THE NOOE AOORESS WHICH WAS TAKEN FROM THE NOOE POOL IS
0239B                   *  PASSEO TO THE BUFFER ONE BYTE AT A TIME.  THE AOORESS IS ALSO
02399                   *  STOREO IN A WORKING REGISTER TEMPORARILY.

02401P 0B7B F6 177F  O        LOAB   AVAILN+1 FETCH THE LSB OF THE NOOE AOORESS
02402P 0B7B E7 07    A        STAB   7,X      INITIALIZE THE BUFFER WITH THE LSB OF THE NOOE AOO
02403P 0B7O F7 1773  O        STAB   TEMPR4+1 STORE THE LSB TEMPORARILY
02404P 0B8O F6 177E  O        LOAB   AVAILN   FETCH THE MSB OF THE NOOE AOORESS
02405P 0B83 E7 06    A        STAB   6,X      INITIALIZE THE BUFFER WITH THE MSB OF THE NOOE AOO
02406P 0B85 F7 1772  O        STAB   TEMPR4   TEMPORARILY STORE THE MSB OF THE NOOE AOORESS
02407P 0B8B A7 00    A        STAA   0,X      VALIOATE THE BUFFER


02409                   *  CHECK TO OETERMINE WHETHER THE PACKET IS AN INFORMATION PACKET.
02410                   *  AN INFORMATION PACKET HAS THE PIGGY-BACK OPTION.  THIS IMPLIES
02411                   *  THAT THERE IS INFORMATION FOR BOTH THE INITIATOR ANO RESPONOER.

02413P 0B8A B1 01    A        CMPA   #1       COMPARE FOR AN INFORMATION PACKET
02414P 0B8C 27 06 0B94        BEQ    INFRES   INFORMATIN PACKET EXISTS IF VALUE IS ZERO


02416                   *  THE PACKET IS OTHER THAN AN INFORMATION PACEKT.
02417                   *  THEREFORE, RETURN THE NOOE TO THE NOOE POOL.

02419P 0B8E FE 1772  O        LOX    TEMPR4   FETCH THE NOOE AOORESS
0242OP 0B91 BO 0076  P RCTURN JSR    PUTNOO   RETURN THE NOOE TO THE NOOE POOL
02421P 0B94 2O AO 0B43 INFRES BRA    RECEIV   REPEAT THE RECEIVER NORMAL LOOP
```

```
02423                    *  THIS CODE ACCESSES THE INITIATOR RESONSE BUFFER.  IT
02424                    *  IS USED TO ISSUE A RR RESPONSE TO THE INITIATOR.  THIS CODE
02425                    *  IS ONLY ACCESSED IF THERE WAS AN I-FRAME AND THE PIGGY-BACK
02426                    *  OPTION WAS IN EFFECT.



0242BP 0B96 CE 175A  D        LDX    #INRESP  FETCH THE PRIMARY BUFFER
02429P 0B99 BD 00BD  P        JSR    WAITBF   WAIT FOR THE BUFFER TO BECOME EMPTY

02431P 0B9C B6 17BF  D        LOAA   NT10FR   FETCH THE ACKNOWLEDGED PACKET SEQUENCE NUMBER
02432P 0B9F A7 05    A        STAA   5,X      INITIALIZE THE BUFFER WITH THE ACKNOWLEDGED NUMBER


02434                    *  THE NODE ADDRESS IS PASSED TO THE PRIMARY BUFFER.  THE
02435                    *  ADDRESS IS NOT USED HERE AND IS ONLY IMPLEMENTED FOR
02436                    *  FUTURE EXPANSION.

0243BP 0BA1 B6 177F  D        LDAA   AVAILN+1 FETCH THE LSB OF THE NODE ADDRESS
02439P 0BA4 F6 177E  D        LDAB   AVAILN   FETCH THE MSB OF THE NODE ADDRESS
0244OP 0BA7 A7 07    A        STAA   7,X      INITIALIZE THE BUFFER WITH THE LSB OF THE NODE ADD
02441P 0BA9 E7 06    A        STAB   6,X      INITIALIZE THE BUFFER WITH THE MSB OF THE NODE ADD

02443P 0BAB 6F 03    A        CLR    3,X      INITIALIZE THE BUFFER WITH A POLL BIT OF ZERO

02445P 0BAD B6 04    A        LDAA   #4       LOAD WITH AN RR RESPONSE CODE
02446P 0BAF A7 01    A        STAA   1,X      INITIALIZE THE BUFFER WITH THE PACKET TYPE
02447P 0BB1 A7 00    A        STAA   0,X      VALIDATE THE BUFFER
0244BP 0BB3 7E 0B43  P        JMP    RECEIV   REAPEAT THE RECEIVER NORMAL LOOP
```

```
02450                    *  DECODE CONTROL WORD

02452                    *  THIS ROUTINE WILL GATHER AN ENTIRE FRAME, DETERMINE ITS
02453                    *  VALIDITY, AND DECODE THE CONTROL WORD.  THE CONTROL WORD IS
02454                    *  THE FOURTH WORD OF THE FRAME.  IT CONTAINS DECODING BITS
02455                    *  AND THE POLL/FINAL BIT.  DEPENDING ON THE PACKET, THE SEQUENCE
02456                    *  NUMBER AND ACKNOWLEDGED SEQUENCE NUMBER MAY ALSO BE DERIVED.



0245B          08B6  P DECODE EQU    *
02459P 08B6 BD 090B  P        JSR    RCPACK   RECEIVE AN ENTIRE FRAME
0246DP 08B9 2B 35 0BFD        BMI    ERREXT   FRAME IS IN ERROR IF VALUE IS NEGATIVE


02462                    *  DECODE THE CONTROL WORD

02464P 08BB FE 177E  D        LDX    AVAILN   FETCH THE FRAME NODE ADDRESS
02465P 08BE E6 01    A        LDAB   1,X      RETRIEVE THE CONTROL WORD

02467P 08C0 17                TBA             ACCUMULATOR B - HOLDING AREA FOR CONTROL WORD
0246BP 08C1 84 10    A        ANDA   #$10     ELIMINATE ALL BITS EXCEPT POLL/FINAL BIT
02469P 08C3 B7 179B  D        STAA   POLFIN   INITIALIZE POLL/FINAL FLAG


02471                    *  DECODE THE (NPROFS) FROM CONTROL WORD.
02472                    *  THIS IS THE SEQUENCE NUMBER OF THE RECEIVED PACKET.

02474P 08C6 17                TBA
02475P 08C7 44                LSRA            ADJUST POSITION OF CONTROL WORD
02476P 08C8 84 07    A        ANDA   #$07     STRIP OFF ALL BUT THE LEAST SIGNIFICANT THREE BITS
02477P 08CA B7 1790  D        STAA   NPROFS   INITIALIZE WITH THE SEQUENCE NUMBER


02479                    *  DECODE FOR (NTIOFR).  DERIVE THE
02480                    *  ACKNOWLEDGED PACKET SEQUENCE NUMBER.

02482P 08CD 17                TBA
02483P 08CE 44                LSRA            ADJUST THE CONTROL WORD TO THE RIGHT
02484P 08CF 44                LSRA
02485P 08D0 44                LSRA
02486P 08D1 44                LSRA
02487P 08D2 44                LSRA
02488P 08D3 B7 178F  D        STAA   NTIOFR   INITIALIZE WITH THE ACKNOWLEDGED SEQUENCE NUMBER
```

```
02490                   *  DECODE THE DECODING BITS TO DETERMINE THE TYPE
02491                   *  OF PACKET WHICH IS BEING SENT.

02493P 08D6 17                  TBA
02494P 08D7 44                  LSRA            ADJUST RIGHT
02495P 08D8 24 17 08F1          BCC     BINTRA  INFORMATION PACKET IF CARRY IS CLEAR
02496P 08DA 84 07     A         ANDA    #$07    STRIP OFF UNUSED BITS
02497P 08DC 27 2A 0908          BEQ     BINTRH  RR RESPONSE PACKET IF VALUE IS ZERO

02499P 08DE 81 01     A         CMPA    #1      COMPARE FOR A DISC COMMAND OR AN UA RESPONSE
02500P 08E0 27 18 08FD          BEQ     BINTRE  TEST VALID IF VALUE IS ZERO

02502P 08E2 81 02     A         CMPA    #2      COMPARE FOR A RNR RESPONSE
02503P 08E4 27 14 08FA          BEQ     BINTRD  TEST POSITIVE IF VALUE IS ZERO

02505P 08E6 81 03     A         CMPA    #3      COMPARE FOR A CMDR COMMAND
02506P 08E8 27 0D 08F7          BEQ     BINTRC  TEST IS POSITIVE IF VALUE IS ZERO

02508P 08EA 81 04     A         CMPA    #4      COMPARE FOR A REJECT RESPONSE
02509P 08EC 27 06 08F4          BEQ     BINTRB  TEST IS POSITIVE IF VALUE IS ZERO



02512                   *  DEFAULT PACKET OF A SARM COMMAND.

02514P 08EE 86 02     A         LDAA    #2      LOAD WITH SARM COMMAND CODE
02515P 08F0 39          ERREXT  RTS
```

```
02517                   *  THE DECODING PROCESS IS COMPLETED.  ASSIGN
02518                   *  THE CODED VALUES TO THE PACKET TYPE.

02520P 08F1 86 01    A 8INTRA LDAA   #1       LOAD WITH INFORMATION COMMAND CODE
02521P 08F3 39                RTS

02523P 08F4 86 06    A BINTRB LDAA   #6       LOAD WITH REJECTION RESPONSE CODE
02524P 08F6 39                RTS

02526P 08F7 86 08    A 8INTRC LDAA   #8       LOAD WITH CMDR RESPONSE CODE
02527P 08F9 39                RTS

02529P 08FA 86 05    A BINTRD LDAA   #5       LOAD WITH RNR RESPONSE CODE
02530P 08FC 39                RTS


02532                   *  FURTHER DECODING IS IN ORDER TO OETERMINE THE DIFFERENCE
02533                   *  BETWEEN A DISC COMMAND AND A UA RESPONSE.

02535P 08FD 17          BINTRE TBA
02536P 08FE 85 20    A         BITA   #$20     STRIP OFF ALL UNNECESSARY 8ITS
02537P 0900 26 03 0905         BNE    BINTRF   DISC COMMAND IF BRANCH NOT TAKEN
02538P 0902 86 07    A         LDAA   #7       LOAD WITH UA RESPONSE CODE
02539P 0904 39                 RTS

02541P 0905 86 03    A BINTRF  LDAA   #3       LOAD WITH DISC COMMAND CODE
02542P 0907 39                 RTS

02544P 0908 86 04    A BINTRH  LDAA   #4       LOAD WITH A RR RESPONSE CODE
02545P 090A 39                 RTS
```

```
02547                   *  RECEIVE ENTIRE FRAME

02549                   *  THIS ROUTINE WILL GATHER AN ENTIRE FRAME INTO A NODE.  THE
02550                   *  ROUTINE WILL SENSE WHEN THE FRAME IS DUE, RETRIEVE A NODE FROM
02551                   *  THE NODE POOL, STRIP OFF TRANSPARENCY BYTES AND MAINTAIN A
02552                   *  CHECKSUM.  THIS ROUTINE CAN ALSO RESTART THE RECEPTION
02553                   *  OF THE FRAME.  IF IT DETECTS THE CONTROL SEQUENCE OF (10,02),
02554                   *  THE OLD FRAME IS DISCARDED AND A NEW FRAME IS ASSUMED.



02556           090B  P RCPACK EQU    *


0255B                   *  THIS ROUTINE LOOKS OIRECTLY AT THE INPUT PORT.  FOR A
02559                   *  (DTE), THE INPUT PORT NUMBER IS (NETWRK).  FOR A
02560                   *  (DCE), THE INPUT PORT NUMBER IS (SUBNET).

02562           0000  A         IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
02563P 090B B6 00     A         LDAA   #NETWRK  PORT NUMBER AT HOST
02564                           ENDC

02566           0000  A         IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
02567                       LDA A #SUBNET  PORT NUMBER - INTERFACE SIDE
0256B                       ENDC

02570P 090D BD 0000   A         JSR    GETCHR   FETCH A CHARACTER FROM THE DESIRED PORT

02572P 0910 C1 10     A         CMPB   #$10     COMPARE FOR A COMMAND WORD
02573P 0912 26 F7 090B          BNE    RCPACK   COMMAND WORD IF VALUE IS ZERO
02574P 0914 BD 0000   A         JSR    GETCHR   FETCH SECOND COMMAND WORD
02575P 0917 C1 02     A         CMPB   #$02     COMPARE FOR AN ASCII (STX)
02576P 0919 26 F0 090B          BNE    RCPACK   TEST POSITIVE IF VALUE IS ZERO
```

```
0257B                   *  FETCH ANO VALIOATE A FRAME

025BO                   *  THIS ROUTINE IS CALLEO WHEN THE CONTROL SEQUENCE OF (HEX 10,02)
025B1                   *  IS RECEIVEO.  THIS ROUTINE HANOLES THE 'PAPER WORK' FOR
025B2                   *  COLLECTING OATA.  IT INITIALIZES THE CHECKSUM, CALLS
025B3                   *  THE OATA COLLECTION ROUTINE, ANO VALIOATES THE FRAME THROUGH
025B4                   *  THE CHECKSUM.

025B6                   *  INITIALIZE THE CHECKSUM.

025BBP 091B C6 12    A RESTAR LOAB   #$12     LOAO WITH THE COMMANO SEQUENCE WOROS
025B9P 091O F7 17B1  O        STAB   RCSUM+1  INITIALIZE THE LSB OF THE CHECKSUM
0259OP 092O 5F                CLRB           CLEAR THE FOLLOWING:
02591P 0921 F7 17BO  O        STAB   RCSUM    MSB OF CHEKSUM
02592P 0924 F7 179F  O        STAB   RCBYTE   BYTE COUNT OF RECIEVER
02593P 0927 C6 BC    A        LOAB   #NOOESZ  LOAO WITH MAXIMUM RECEPTION COUNT
02594P 0929 F7 17AB  O        STAB   RCLIMT   INITIALIZE VARIABLE WITH MAXIMUM RECEPTION COUNT

02596P 092C BO 0057  P        JSR    GETNOO   FETCH FRAME NOOE
02597P 092F FF 177E  O        STX    AVAILN   STORE THE NOOE AORRESS POINTER (STATIC)
0259BP 0932 FF 1772  O        STX    TEMPR4   STORE THE NOOE AOORESS POINTER (OYNAMIC)
02599P 0935 BO 0960  P        JSR    RCFRME   RECEIVE FRAME EXCEPT CHECKSUM
026OOP 093B 2B 1A 0954        BMI    PKERR    RESTART SENSEO IF VALUE IS NEGATIVE


02602                   *  THE FRAME HAS BEEN RECEIVEO.  CHECK THE VALIOITY OF THE
02603                   *  FRAME BY COMPARING THE RECEIVEO CHECKSUM WITH THE CALCULATEO
02604                   *  CHECKSUM.  THE RECEIVEO VALUE IS IN ITS NEGATIVE FORM.
02605                   *  THEREFORE, AOOING THE TWO VALUES SHOULO PROOUCE A RESULT
02606                   *  OF ZERO IF THE FRAME HAO A FLAWLESS TRASNMISSION.

026OBP 093A BO OOOO  A        JSR    GETCHR   FETCH THE MSB OF THE RECEIVEO CHECKSUM
02609P 093O 37                PSHB
0261OP 093E BO OOOO  A        JSR    GETCHR   FETCH THE LSB OF THE RECEIVEO CHECKSUM
02611P 0941 17                TBA
02612P 0942 33                PULB
02613P 0943 BB 17B1  O        AOOA   RCSUM+1  AOO THE LSB OF THE CHECKSUMS
02614P 0946 F9 17BO  O        AOCB   RCSUM    AOO THE MSB OF THE CHECKSUMS
02615P 0949 F7 1772  O        STAB   TEMPR4   STORE FOR JUST A MOMENT
02616P 094C BA 1772  O        ORAA   TEMPR4   SUPERIMPSE THE MSB WITH THE LSB
02617P 094F 27 02 0953        BEQ    *+4      FLAWLESS TRANSMISSION IF VALUE IS ZERO
0261BP 0951 B6 BO    A        LOAA   #$BO     ERROR IN TRANSMISSION   SIGNAL APPROPRIATELY
02619P 0953 39                RTS
```

```
02621                   *  RESTART TRANSMISSION

02623                   *  A RETRANSMISSION SEQUENCE HAS BEEN DETECTED.  IT HAS THE CODE
02624                   *  OF A (HEX 10,02).  THIS SEQUENCE IMPLIES THAT THE FRAME
02625                   *  IS TO BE REJECTED AND START A NEW FRAME RECEPTION.  IN THIS
02626                   *  CASE, CONTROL WILL CYCLE BACK TO THE (RESTAR) ROUTINE.  ON THE
02627                   *  OTHERHAND, A GENERAL ERROR COULD HAVE BEEN SENSED.  IF THIS
02628                   *  IS THE CASE, THE STARTING SEQUENCE HAS NOT BEEN DETECTED.
02629                   *  CYCLE BACK TO THE (RCPACK) ROUTINE.


02631P 0954 FE 177E  0 PKERR  LDX    AVAILN   FETCH THE NODE ADDRESS
02632P 0957 BD 0076  P        JSR    PUTNOD   RETURN THE NODE TO THE NODE POOL
02633P 095A C5 04    A        BITB   #04      TEST WHETHER A GENERAL ERROR HAS OCCURRED
02634P 095C 27 AD 0908        BEQ    RCPACK   TEST IS POSITIVE IF VALUE IS ZERO
02635P 095E 20 BB 091B        BRA    RESTAR   CYCLE BACK TO FETCH A NEW FRAME
```

```
02637                   *  RECEIVE FRAME

02639                   *  THIS ROUTINE WILL COLLECT THE BYTES THAT MAKE UP THE FRAME.
02640                   *  THE BYTES ARE STORED IN THE NODE WHOSE ADDRESS IN PASSED IN THE
02641                   *  INDEX REGISTER.  THE TRANSPARENCY BYTES ARE STRIPPED OFF (IF
02642                   *  ANY) AND A CHECKSUM IS MAINTAINED.  THE BYTE COUNT WILL
02643                   *  HAVE A MAXIMUM VALUE OF THE NODE SIZE.  IF IT IS EXCEEDED, A
02644                   *  GENERAL ERROR WILL BE PRODUCED.  ALSO, IF THE ROUTINE DETECTS
02645                   *  A CONTROL COMMAND SEQUENCE OF (HEX 10,02), AN ERROR WILL BE
02646                   *  GENERATED.  THIS IS KNOWN AS A RESTART ERROR.



0264B           0960  P RCFRME EQU    *
02649P 0960 BD 0000   A         JSR    GETCHR   FETCH A FRAME BYTE
0265OP 0963 BD 09BA   P         JSR    PCPOST   INCREMENT COUNT AND ADD TO CHECKSUM


02652                   *  THE RECEIVED BYTE MUST BE TESTED FOR A POSSIBLE TRANSPARENCY
02653                   *  BYTE.  THIS IS A BYTE WHICH IS ADDED TO THE FRAME TO DISTINGUISH
02654                   *  IT FROM A DATA BYTE.  THE BYTE IN QUESTION IS AN ASCII (HEX 10).

02656P 0966 C1 10     A         CMPB   #$10     COMPARE FOR POSSIBLE TRANSPARENCY BYTE
02657P 096B 26 F6 0960          BNE    RCFRME   NOT A TRANSPARENCY BYTE IF VALUE IS NON-ZERO

02659                   *  POSSIBLE TRANSPARENCY BYTE

02661P 096A BD 0000   A         JSR    GETCHR   FETCH NEXT FRAME BYTE
02662P 096D C1 02     A         CMPB   #02      COMPARE FOR A RESTART COMMAND
02663P 096F 27 16 09B7          BEQ    RESTRT   RESTART COMMAND IF VALUE IS ZERO
02664P 0971 C1 10     A         CMPB   #$10     COMPARE FOR TRANSPARENCY BYTE
02665P 0973 26 05 097A          BNE    *+7      TEST IS VALID IF VALUE IS ZERO
02666P 0975 BD 099E   P         JSR    RECSUM   ADD TRANSPARENCY BYTE TO CHECKSUM, INCREMENT COUNT
02667P 097B 20 E6 0960          BRA    RCFRME   CYCLE AND FETCH NEXT FRAME BYTE


02669                   *  ENTRY POINT FOR A NON-TRANSPARENCY FRAME BYTE.

02671P 097A BD 098A   P         JSR    PCPOST   ADD BYTE TO CHECKSUM AND INCREMENT BYTE COUNTER
02672P 097D C1 03     A         CMPB   #$03     COMPARE FOR FRAME TERMINATOR
02673P 097F 26 03 09B4          BNE    EXTABT   FRAME TERMINATOR IF VALUE IS ZERO
02674P 09B1 C6 01     A         LDAB   #$01     VALID RECEPTION
02675P 09B3 39                  RTS

02677P 09B4 C6 B2     A EXTABT  LDAB   #$82     ERROR   GENERAL
0267BP 09B6 39                  RTS

026BOP 09B7 C6 B4     A RESTRT  LDAB   #$B4     ERROR - RESTART
026B1P 09B9 39                  RTS
```

```
026B3                   *  POST RECEPTION

026B5                   *  THIS ROUTINE WILL MAINTAIN A RUNNING TOTAL ON THE CHECKSUM.
026B6                   *  THE BYTE IS ADDED TO A SIXTEEN BIT WORKING REGISTER.  THIS
026B7                   *  ROUTINE ALSO WILL MAINTAIN THE RECEIVER BYTE COUNTER.



026B9           09BA  P PCPOST EQU    *
02690P 09BA 70 17A8  0         TST    RCLIMT   TEST FOR MAXIMUM BYTE COUNT
02691P 09BD 27 1C 09AB         BEQ    PSTEX    TEST IS POSITIVE IF VALUE IS ZERO
02692P 09BF 7A 17A8  0         DEC    RCLIMT   DECREMENT THE LIMIT BY ONE

02694P 09C2 FE 1772  0         LDX    TEMPR4   FETCH THE DYNAMIC NODE ADDRESS
02695P 09C5 E7 00    A         STAB   0,X      STORE THE FRAME BYTE INTO NODE
02696P 09C7 08                 INX             INCREMENT NODE ADDRESS
02697P 09C8 FF 1772  0         STX    TEMPR4   UPDATE NODE ADDRESS BYTE POINTER

02699P 09CB 7C 179F  0         INC    RCBYTE   INCREMENT THE BYTE COUNT BY ONE

02701P 09CE 37          RECSUM PSHB
02702P 09CF FB 17B1  0         ADDB   RCSUM+1  ADD DATA BYTE TO LSB OF CHECKSUM
02703P 09D2 F7 17B1  0         STAB   RCSUM+1  UPDATE LSB OF CHECKSUM
02704P 09D5 24 03 09DA         BCC    *+5      OVERFLOW (CARRY) IF BRANCH NOT TAKEN
02705P 09D7 7C 17B0  0         INC    RCSUM    ADJUST MSB OF CHECKSUM ACCORDINGLY
02706P 09DA 33                 PULB
02707P 09DB 39          PSTEX  RTS
```

```
02709                   ************************************************************************
02710                   *                                                                      *
02711                   *    INPUT                                                             *
02712                   *                                                                      *
02713                   *    THIS PROCESS WILL SIMULATE A USER.  AN ASCII PACKET IS BUILT      *
02714                   *    USING 127 CHARACTERS.  THIS WILL REPRESENT A PACKET               *
02715                   *    SENT TO THE INITIATOR FOR TRANSMISSION BY THE HOST USER.          *
02716                   *    THE ROUTINE WILL INSERT PACKETS AS FAST AS IT CAN.  THIS WILL     *
02717                   *    SIMULATE A HIGH BAUD SITUATION OR TRYING TO SATURATE THE          *
0271B                   *    LINK OF RECOMMENDATTION X.25.  NORMAL OPERATONS REQUIRE THAT      *
02719                   *    THIS PROCESS BE SUBSTITUTED FOR THE USER TASK.                    *
02720                   *                                                                      *
02721                   ************************************************************************



02723           0000  A         IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
02724           09AC  P TKPC4   EQU    *
02725                           ENDC

02727           0000  A         IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
0272B                   TKPCB EQU *
02729                     ENDC
02730P 09AC BD 0057   P INPUTS  JSR    GETNOD   FETCH A NODE FROM THE NODE POOL
02731P 09AF FF 176C   D         STX    TEMPR1   SAVE THE NODE ADDRESS IN THE WORKING REGISTER
02732P 09B2 FF 177C   D         STX    INADDR   SAVE THE NODE ADDRESS IN THE STATIC REGISTER
02733P 09B5 7F 179C   D         CLR    INCNT    CLEAR BYTE COUNT
02734P 09BB BD 0905   P         JSR    RECDAT   BUILD A PACKET
02735P 09BB CE 1752   D         LDX    #INDATA  FETCH ADDRESS OF PRIMARY POINTER
02736P 09BE BD 00BD   P         JSR    WAITBF   WAIT FOR BUFFER TO BECOME EMPTY


0273B                   *  THE BUFFER IS READY AND THE NODE IS FULL.
02739                   *  INITIALIZE THE PRIMARY BUFFER.

02741P 09C1 B6 177D   D         LDAA   INADDR+1 FETCH THE LSB OF THE NODE ADDRESS
02742P 09C4 F6 177C   D         LDAB   INADDR   FETCH THE MSB OF THE NODE ADDRESS
02743P 09C7 A7 03     A         STAA   3,X      INITIALIZE BUFFER WITH THE LSB OF THE NODE ADDRESS
02744P 09C9 E7 02     A         STAB   2,X      INITIALIZE BUFFER WITH THE MSB OF THE NODE ADDRESS
02745P 09CB B6 179C   D         LDAA   INCNT    FETCH THE BYTE COUNT
02746P 09CE A7 01     A         STAA   1,X      INITIALIZE THE BUFFER WITH THE INPUT BYTE COUNT
02747P 09D0 E7 00     A         STAB   0,X      VALIDATE THE BUFFER

02749P 09D2 3F                  SWI            RETURN CONTROL TO SCHEDULER
02750P 09D3 20 07 09AC          BRA    INPUTS   CYCLE AND BUILD ANOTHER PACKET
```

```
02752                    *  BUILD A PACKET

02754                    *  THIS ROUTINE WILL BUILD A PACKET WHICH WILL BE SENT TO THE
02755                    *  INITIATOR FOR TRANSMISSION.  THE PACKET IS BUILT USING ASCII
02756                    *  CHARACTERS.  THE MAXIMUM LENGTH OF THE PACKET IS 128 BYTES.
02757                    *  THE BYTE COUNT WILL BE RETURN IN (INCNT).



02759           0905  P RECOAT EQU    *
02760P 0905 86 17A7  0         LOAA   DATA     FETCH THE DATA
02761P 0908 FE 176C  0         LOX    TEMPR1   FETCH THE NEXT DATA LOCATION
02762P 0908 81 80    A         CMPA   #$80     COMPARE FOR MAXIMUM ASCII VALUE
02763P 0900 26 02 09E1         BNE    *+4      RESET ASCII COUNT IF VALUE IS ZERO

02765P 090F 86 2E    A         LOAA   #$2E     RESET ASCII COUNT
02766P 09E1 A7 00    A         STAA   0,X      STORE DATA IN NODE
02767P 09E3 4C                 INCA           INCREMENT ASCII COUNT
02768P 09E4 B7 17A7  0         STAA   DATA     STORE ASCII VALUE
02769P 09E7 08                 INX            INCREMENT NODE POINTER
02770P 09E8 FF 176C  0         STX    TEMPR1   UPDATE NODE POINTER IN WORKING REGISTER
02771P 09EB 7C 179C  0         INC    INCNT    INCREMENT BYTE COUNT BY ONE
02772P 09EE 86 179C  0         LOAA   INCNT    FETCH BYTE COUNT

02774           0000  A         IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
02775P 09F1 81 7F    A         CMPA   #$7F     COMPARE FOR MAXIMUM ASCII COUNT
02776                          ENDC

02778           0000  A         IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
02779                      CMP A #$40  COMPARE WITH MAXIMUM COUNT
02780                      ENDC

02782P 09F3 26 E0 0905         BNE    RECOAT   CYCLE AND CONTINUE IF NOT MAXIMUM VALUE
02783P 09F5 39                 RTS
```

```
02785                  ************************************************************************
02786                  *                                                                      *
02787                  *    OUTPUT                                                            *
02788                  *                                                                      *
02789                  *    THIS PROCESS SIMULATES A USER.  IT IS USED TO OUTPUT A PACKET     *
02790                  *    TO AN OUTPUT DEVICE.  THE DATA PACKETS ARE CONSTRUCTED USING      *
02791                  *    ASCII CHARACTERS.  THIS ROUTINE WILL WAIT FOR THE PRIMARY         *
02792                  *    BUFFER TO BECOME VALID.  THE BUFFERS ARE SWAPPED AND THE DATA     *
02793                  *    FROM THE NODE IS SENT TO THE SCREEN.  WHEN THE NODE IS EMPIED,    *
02794                  *    IT IS RETURNED TO THE NODE POOL.  THE NORMAL SITUATION            *
02795                  *    CALLS FOR THIS PROCESS TO BE SUBSTITUTED FOR A USER PROCESS.      *
02796                  *    ONE WHERE THE DATA IS EXTRACTED FROM THE BUFFERS.                 *
02797                  *                                                                      *
02798                  ************************************************************************



02800           0000  A        IFEQ   CHANGE   ASSEMBLE FOR (DTE) IF VALUE IS ZERO
02801           09F6  P TKPC5  EQU    *
02802                          ENDC

02804           0000  A        IFNE   CHANGE   ASSEMBLE FOR (DCE) IF VALUE IS NON-ZERO
02805                    TKPCC EQU *
02806                      ENDC

02808P 09F6 BD 0A32  P OUTPUT JSR    WTSCDT   WAIT FOR BUFFER AND EXCHANGE POINTERS


02810                    *  RETRIEVE INFORMATION FROM THE BUFFER AND DEVALIDATE IT.
02811                    *  CALL ON ROUTINE TO ISSUE DATA TO THE OUTPUT DEVICE AND
02812                    *  RETURN THE NODE TO THE NODE POOL.

02814P 09F9 EE 02    A          LDX    2,X      FETCH THE PACKET ADDRESS
02815P 09FB FF 1776  D          STX    TEMPR7   STORE IN THE WORKING REGISTER
02816P 09FE FF 1782  D          STX    OUTADR   STORE IN THE STATIC REGISTER
02817P 0A01 FE 1768  D          LDX    STDBY6   FETCH BUFFER SECONDARY ADDRESS
02818P 0A04 A6 01    A          LDAA   1,X      RETRIEVE BYTE COUNT
02819P 0A06 6F 00    A          CLR    0,X      DEVALIDATE THE BUFFER
02820P 0A08 B7 1790  D          STAA   OUTCNT   OUTPUT BYTE COUNT
02821P 0A0B BD 0A16  P          JSR    ISSDAT   ISSUE DATA TO OUTPUT DEVICE
02822P 0A0E FE 1782  D          LDX    OUTADR   FETCH NODE ADDRESS
02823P 0A11 BD 0D76  P          JSR    PUTNOD   RETURN NODE TO NODE POOL
02824P 0A14 20 E0 09F6          BRA    OUTPUT   CYCLE BACK AND WAIT FOR NEXT PACKET
```

```
02826                    *  ISSUE OATA

02828                    *  THIS ROUTINE WILL ISSUE ALL OF THE OATA CONTAINEO IN ONE PACKET
02829                    *  TO THE OUTPUT OEVICE.  IT IS ASSUMEO THAT THE PACKET IS MAOE UP
02830                    *  OF ASCII CHARACTERS ANO THE OUTPUT OEVICE IS A CRT.



02832          0A16  P ISSOAT EQU    *
02833P 0A16 FE 1776  0        LOX    TEMPR7   FETCH THE NOOE AOORESS
02834P 0A19 08                INX             AOJUST POINTER TO FIRST OATA 8YTE
02835P 0A1A 08                INX
02836P 0A18 FF 1776  0        STX    TEMPR7   UPOATE THE OATA POINTER
02837P 0A1E FE 1776  0 ISOAT2 LOX    TEMPR7   FETCH THE NEXT OATA POSITION IN THE NOOE
02838P 0A21 E6 00    A        LOA8   0,X      FETCH THE OATA FROM THE NOOE
02839P 0A23 08                INX             INCREMENT POINTER TO NEXT 8YTE
02840P 0A24 FF 1776  0        STX    TEMPR7   UPOATE THE 8YTE POINTER

02842          0000  A        IFEQ   CHANGE   ASSMEL8E FOR (OTE) IF VALUE IS ZERO
02843P 0A27 86 02    A        LOAA   #URACIA  HOST ACIA
02844                         ENOC

02846          0000  A        IFNE   CHANGE   ASSEM8LE FOR (OCE) IF VALUE IS ZERO
02847                     LOA A #S8ACIA  NETWORK USER PORT
02848                     ENOC

02850P 0A29 80 0000  A        JSR    PUTCHR   ISSUE OATA TO OUTPUT OEVICE
02851P 0A2C 7A 1790  0        OEC    OUTCNT   OECREMENT OUTPUT 8YTE COUNT
02852P 0A2F 26 E0 0A1E        8NE    ISOAT2   REPEAT LOOP UNTIL NOOE IS EXHAUSTEO
02853P 0A31 39                RTS
```

```
02855                   *  WAIT FOR BUFFER (OUTPUT)

02857                   *  THIS ROUTINE WILL CYCLE WAITING FOR THE PRIMARY BUFFER OF
02858                   *  (OUTDAT) TO BECOME VALID.  WHEN IT IS NOT, CONTROL WILL
02859                   *  BE PASS TO THE SCHEDULER.  THE CYCLE WILL LAST INDEFINITELY.
02860                   *  THE BUFFERS ARE SWAPPED AND THE ROUTINE EXITS WHEN THE BUFER
02861                   *  IS VALID.


02863          0A32  P WTSCOT EQU    *
02864P 0A32 FE 1766  D        LDX    OUTDAT   FETCH THE PRIMARY BYTE POINTER
02865P 0A35 A6 00    A        LDAA   0,X      INTERROGATE THE STATUS WORD
02866P 0A37 26 03 0A3C        BNE    *+5      BUFFER IS EMPTY IF THE VALUE IS ZERO
02867P 0A39 3F                SWI             RETURN CONTROL TO THE SCHEDULER
02868P 0A3A 20 F6 0A32        BRA    WTSCOT   REPEAT WAIT LOOP


02870                   *  THE BUFFER IS VALID.  SWAP POINTERS.

02872P 0A3C 0F                SEI             ENTERING CRITICAL REGION
02873P 0A3D 7C 0000  A        INC    CRTFLG
02874P 0A40 FF 1776  D        STX    TEMPR7
02875P 0A43 FE 1768  D        LDX    STDBY6
02876P 0A46 FF 1766  D        STX    OUTDAT
02877P 0A49 FE 1776  D        LDX    TEMPR7
02878P 0A4C FF 1768  D        STX    STDBY6
02879P 0A4F 7A 0000  A        DEC    CRTFLG   EXITING CRITICAL REGION
02880P 0A52 0E                CLI
02881P 0A53 39                RTS

02883                         END
TOTAL ERRORS 00000
```

```
00001  ****************************************************************************
00002  ****************************************************************************
00003  **                                                                        **
00004  **                                                                        **
00005  **    MULTI-TASKING MONITOR SYSTEM                                        **
00006  **                                                                        **
00007  **    THE PURPOSE OF THIS MONITOR IS TO EXECUTE THE PROTOCOL              **
00008  **    RECOMMENDATION X.25.  IT IS A GENERAL PURPOSE MONITOR               **
00009  **    AND, THEREFORE, MAY BE USED AS SUCH.  IT SUPPORTS UP TO             **
00010  **    FOURTEEN CONCURRENT PROCESS.  THERE IS ONE ADDITIONAL               **
00011  **    PRIORITY PROCESS.  IT ALSO CAN HANDLE UP TO FOUR DEVICES.           **
00012  **    THE SERVICE ROUTINE FOR AN ACIA AND PIA ARE SUPPLIED,               **
00013  **    (THE PIA SERVICE ROUTINE IS UNCHECKED).  THIS MONITOR               **
00014  **    ALSO SUPPORTS BUFFERED I/O.  THERE ARE SIXTEEN BYTES                **
00015  **    PER QUEUE AND TWO QUEUES PER DEVICE.                                **
00016  **                                                                        **
00017  **    ALL DEVICE INFORMATION IS CONTAINED IN TWO TABLES.                  **
00018  **    (IRQTBL) CONTAINS PERMANENT INFORMATION AND IS LOCATED IN           **
00019  **    ROM.  (VARQUE) CONTAINS VARIABLE INFORMATION ON EACH DEVICE.        **
00020  **    BOTH ARE INTERNALLY LINKED THROUGH POINTERS.                        **
00021  **    INFORMATION ON PROCESSES MAY BE FOUND BETWEEN THREE TABLES.         **
00022  **    (RESTSK) CONTAINS THE STATIC STACK POINTERS AND STATUS              **
00023  **    INFORMATION.  THIS TABLE IS IN ROM.  (PROCSS) CONTAINS THE          **
00024  **    DYNAMIC STACK POINTERS AND CORRESPONDING STATUS                     **
00025  **    INFORMATION.  (STCKBL) IS THE STACK TABLE FOR ALL FIFTEEN           **
00026  **    TASKS.  EACH TASK HAS A STACK SIZE OF 64 BYTES.                     **
00027  **                                                                        **
00028  **                                                                        **
00029  ****************************************************************************
00030  ****************************************************************************


00032  ****************************************************************************
00033  ****************************************************************************
00034  **                                                                        **
00035  **                                                                        **
00036  **    WRITTEN BY:     LOUIS D. NYERGES                                    **
00037  **                    CANDIDATE FOR THE MASTERS DEGREE                    **
00038  **                    COMPUTER ENGINEERING                                **
00039  **                    DR. ROY S. CZERNIKOWSKI, ADVISOR                    **
00040  **                                                                        **
00041  **    SYSTEM:         MOTOROLA EXORCISER,  M6800 BASED                    **
00042  **    MEMORY                                                              **
00043  **      REQUIREMENTS: 32K RAM                                             **
00044  **    HARDWARE                                                            **
00045  **      REQUIREMENTS: I/O BOARD CONSISTING OF FOUR ACIA'S                 **
00046  **                    REAL TIME CLOCK OF 2152 HERTZ                       **
00047  **    ADDITIONAL                                                          **
00048  **      INFORMATION:  PLEASE REFERENCE THESIS                             **
00049  **                    "AN INVESTIGATION AND IMPLEMENTATION                **
00050  **                     OF THE C.C.I.T.T. RECOMMENDATION X.25              **
00051  **                     ON A M6800 PILOT BASED NETWORK"                    **
00052  **                                                                        **
00053  **                                                                        **
00054  ****************************************************************************
00055  ****************************************************************************
```

```
00057                          NAM    THESIS:MONITOR



00059                  *  ASSEMBLER DIRECTIVES

00061                  *  OPT (OPTIONS)
00062                  *    REL   - RELOCATABLE CODE
00063                  *    PAGE  - LINES PER PAGE
00064                  *    LLEN  - PLATTEN WIDTH
00065                  *    OBJ   - CREATE OBJECT CODE
00066                  *    MEX   - PRINT MACRO EXPANSIONS
00067                  *    MD    - LIST MACRO DEFINITIONS
0006B                  *    MC    - LIST MACRO CALLS
00069                  *    GEN   - LIST MULTIPLE LINES OF FCC,FCB,FDB
00070                  *  TTL   - PRINT TITLE AT TOP OF EACH PAGE
00071                  *  IDNT  - OBJECT CODE IDENTIFIER
00072                  *  XREF  - EXTERNAL REFERENCES
00073                  *            (LABELS DEFINED IN THIS CODE)
00074                  *  XDEF  - EXTERNAL DEFINITIONS
00075                  *            (LABELS DEFINED ELSEWHERE)


00077                          OPT    REL
00078                          OPT    PAGE=58,LLEN=96
00079                          OPT    OBJ
00080                          OPT    MEX,MD,MC,GEN
00081                          TTL    *** RECOMMENDATION X.25 PROTOCOL ***
00082                          IDNT   L.D. NYERGES - SUMMER, '80. :MONITOR
00083                          XREF   PSCT:TKPC0,TKPC1,TKPC2,TKPC3,TKPCM
00084                          XREF   PSCT:TKPC4,TKPC5,INTDCE,INTDTE
00085                          XREF   PSCT:TKPC6,TKPC7,TKPC8,TKPC9
00086                          XREF   PSCT:TKPCA,TKPCB,TKPCC,TKPCD
00087                          XDEF   GETCHR,PUTCHR,XCHSTS,CRTFLG
00088                          XDEF   IRQSVC,SWISVC,NMISVC,RSTSVC
```

```
00090                   *  ABSOLUTE CODE

00092                   *  THE LABELS WHICH APPEAR BELOW WILL BE ASSIGNEO THEIR
00093                   *  STATED VALUES AT ASSEMBLY TIME.  THEY DEFINE THE SYSTEM
00094                   *  AS IT IS SUBMITTED.  CHANGES TO SOME OF THESE PARAMETERS WILL
00095                   *  REQUIRE OTHER MOOIFICATIONS ELSEWHERE IN THE PROGRAM.
00096                   *  IT SHOULD BE MAOE AWARE THAT ARBITRARY CHANGING TO THE
00097                   *  VALUES BELOW COULD RESULT IN A NONFUNCTIONAL MONITOR.


00099A 0000                     ASCT

00101           000E  A NTASK  EQU    14        NUMBER OF CONCURRENT PROCESSES
00102           0010  A BLOCK  EQU    16        SIZE OF EACH QUEUE IN BYTES
00103           0004  A DEVICE EQU    4         NUMBER OF SUPPORTEO DEVICES
00104           00D5  A ACNTRL EQU    $D5       ACIA CONTROL REGISTER WORD
00105           ECC0  A NETWRK EQU    $ECC0     NETWORK AODRESS PORT FROM OTE TO DCE
00106           ECD0  A RTSD00 EQU    $ECD0     REQUEST-TO-SENO (RTS) CONTROL PORT FROM DTE TO DCE
00107           ECC4  A SUBNET EQU    $ECC4     NETWORK ADDRESS PORT FROM DCE TO DTE
0010B           ECD1  A RTSD01 EQU    $ECD1     REQUEST-TO-SEND (RTS) CONTROL PORT FROM DCE TO DTE
00109           FCF4  A URACIA EQU    $FCF4     USER ADORESS PORT
00110           0000  A RTS003 EQU    $0000     REQUEST-TO-SEND (RTS) PORT.  NOT USED HERE.
00111           ECCB  A SBACIA EQU    $ECCB     NETWORK ADDRESS PORT
00112           ECD2  A RTSD02 EQU    $EC02     REQUEST-TO-SEND (RTS) PORT.  NOT USED HERE.
00113           9000  A USRPIA EQU    $9000     USER ADDRESS PORT (PIA).  NOT USED HERE.
```

```
00115                    *  BASE PAGE

00117                    *  THIS SECTION DEFINES THE DATA STRUCTURE OF THE BASE PAGE.
00118                    *  CHANGES TO THE PRECEDING PARAMETERS WILL CREATE A
00119                    *  STRUCTUAL CHANGE HERE.
00120                    *  NOTE:  DUE TO THE SIZE OF THE BASE PAGE (256 BYTES),  VERY
00121                    *         LITTLE EXPANSION IS POSSIBLE.  THE MONITOR USES ALL
00122                    *         OF THE BASE PAGE.  THERE ARE TWO VARIABLES IN THE OSCT
00123                    *         SECTION OF CODE THAT COULD NOT BE PLACED HERE.


00125B 0000                       BSCT

00127B 0000  0080  A QUETBL RMB   DEVICE*BLOCK*2 BUFFER I/O FOR DEVICES
00128B 0080  0038  A VARQUE RMB   DEVICE*14 THE VARIABLE TABLE OF (IRQTBL)
00129B 00B8  002A  A PROCSS RMB   NTASK*3  THE STACK POINTER DYMANIC TABLE
00130B 00E2  0003  A PRITSK RMB   3        CONTINUATION OF (PROCSS),  PRIORITY STACK POINTER
00131B 00E5  0008  A PARMPT RMB   DEVICE*2 ATTRIBUTE TABLE FOR DEVICES
00132B 00E0  0002  A TSKPTR RMB   2        POINTER TO CURRENT TASK [POINTS INTO (PROCSS)]


00134                    *  THE NEXT FIVE VARIABLES ARE DEVICE AND TASK DEPENDENT.
00135                    *  THEY ARE PUSHED ON THE THE STACK, IF FOR SOME REASON, THE
00136                    *  MONITOR HAS TO CHANGE TO A NEW PROCESS OR TO A NEW DEVICE.

00138B 00EF  0002  A IRQREC RMB   2        POINTER TO CURRENT DEVICE LINE IN (IRQTBL)
00139B 00F1  0002  A IRQVAL RMB   2        POINTER TO CURRENT QUEUE IN DEVICE LINE
00140B 00F3  0002  A QUEREC RMB   2        POINTER TO CURRENT DEVICE LINE IN (VARQUE)
00141B 00F5  0001  A DATA   RMB   1        THE CURRENT WORD TO BE TRANSMITTED OR RECEIVED
00142B 00F6  0001  A CRTFLG RMB   1        FLAG SET WHEN A TASK IS IN A CRITICAL REGION


00144B 00F7  0001  A SUBVAL RMB   1        THE NUMBER OF ELEMENTARY TICKS IN A PRIMARY TICK
00145B 00F8  0001  A PRIVAL RMB   1        THE NUM8ER OF PRIMARY TICKS IN A TIME SLICE


00147                    *  DOUBLE DEFINED DEFINITIONS

00149                    *  THE SIXTEEN BIT POINTERS (SOURCE) AND (DESTIN) ARE USED TO
00150                    *  INITIALIZE RAM BEFORE THE FIRST TASK IS EXECUTED.  INFORMATION
00151                    *  IS TRANSFERRED FROM ROM INTO RAM.  THIS IS THE ONLY TIME THAT
00152                    *  THESE TWO VARIABLES ARE USED.

00154        00F9  B SOURC  EQU   *
00155B 00F9  0001  A INTFLG RMB   1        SET WHEN SEARCHING FOR A NEW TASK
00156B 00FA  0001  A SKPFLG RMB   1        SET WHEN A TASK CHANGE IS CALL FROM A SWI
00157        00FB  B DESTIN EQU   *
00158B 00FB  0002  A TEMP   RMB   2        TEMPORARY LOCATION USED WITH THE ROUTINE (XCHSTS)
00159        00F0  B INTEND EQU   *        END MARKER FOR CLEARING RAM
00160B 00F0  0001  A COUNT  RMB   1        COUNTER USED WITH (IRQSVC) AND (LOADR)
00161B 00FE  0001  A HOLDIT RMB   1        SWITCH TO DISABLE NMI FROM (NMIENT)
```

```
00163                   *  TABLE: STCKBL

00165                   *  THIS TABLE CONTAINS THE STACK AREA FOR EACH PROCESS.  EACH
00166                   *  STACK AREA CONTAINS 64 BYTES.  THE STACKING ORDER IS OBSERVED
00167                   *  HERE.  THEREFORE, INITIALLY, THE STACK POINTERS FOUND IN
00168                   *  (PROCSS) WILL POINT TO THE LABELS OF THIS TABLE.  THE ACTUAL
00169                   *  ADDRESS OF THE PROCESS IS BYTE SIX AND SEVEN UP FROM
00170                   *  THE LABEL.  INITIALLY, ALL REGISTERS ARE ZERO.


00172D 0000                     OSCT

00174           0000  D STCKBL EQU   *
00175D 0000     003B  A        RMB   56        TASK D STACK AREA
00176           003B  D TSKNMD EQU   *
00177D 003B     0006  A        RMB   6         TASK D REGISTER AREA
00178D 003E     0000  A        FDB   TKPCD     STARTING ADDRESS FOR PROCESS D

00180D 0040     003B  A        RMB   56        TASK 1 STACK AREA
001B1           0078  D TSKNM1 EQU   *
00182D 0078     0006  A        RMB   6         TASK 1 REGISTER AREA
00183D 007E     0000  A        FDB   TKPC1     STARTING ADDRESS OF PROCESS 1

001B5D 00BD     0038  A        RMB   56        TASK 2 STACK AREA
00186           00BB  D TSKNM2 EQU   *
00187D 00BB     0006  A        RMB   6         TASK 2 REGISTER AREA
00188D 00BE     0000  A        FDB   TKPC2     STARTING ADDRESS FOR PROCESS 2

00190D 00CD     003B  A        RMB   56        TASK 3 STACK AREA
00191           00F8  D TSKNM3 EQU   *
00192D 00F8     0006  A        RMB   6         TASK 3 REGISTER AREA
00193D 00FE     0000  A        FDB   TKPC3     STARTING ADDRESS FOR PROCESS 3

00195D 0100     0038  A        RMB   56        TASK 4 STACK AREA
00196           0138  D TSKNM4 EQU   *
00197D 013B     0006  A        RMB   6         TASK 4 REGISTER AREA
00198D 013E     0000  A        FDB   TKPC4     STARTING ADDRESS FOR PROCESS 4

002000 0140     0038  A        RMB   56        TASK 5 STACK AREA
00201           0178  D TSKNM5 EQU   *
002020 017B     0006  A        RMB   6         TASK 5 REGISTER AREA
002030 017E     0000  A        FDB   TKPC5     STARTING ADDRESS FOR PROCESS 5

002050 018D     003B  A        RMB   56        TASK 6 STACK AREA
00206           01B8  D TSKNM6 EQU   *
002070 01BB     0006  A        RMB   6         TASK 6 REGISTER AREA
0020B0 01BE     0000  A        FDB   TKPC6     STARTING ADDRESS OF PROCESS 6
```

```
00210D 01C0   003B  A         RMB   56       TASK 7 STACK AREA
00211         01FB  D TSKNM7  EQU   *
00212D 01FB   0006  A         RMB   6        TASK 7 REGISTER AREA
00213D 01FE   0000  A         FDB   TKPC7    STARTING ADDRESS FOR PROCESS 7

00215D 0200   003B  A         RMB   56       TASK B STACK AREA
00216         023B  D TSKNMB  EQU   *
00217D 023B   0006  A         RMB   6        TASK B REGISTER AREA
0021BD 023E   0000  A         FDB   TKPCB    STARTING AODRESS FOR PROCESS B

00220D 0240   003B  A         RMB   56       TASK 9 STACK AREA
00221         027B  D TSKNM9  EQU   *
00222D 027B   0006  A         RMB   6        TASK 9 REGISTER AREA
00223D 027E   0000  A         FDB   TKPC9    STARTING ADDRESS FOR PROCESS 9

00225D 0280   003B  A         RMB   56       TASK 10 STACK AREA
00226         02BB  D TSKNMA  EQU   *
00227D 02BB   0006  A         RMB   6        TASK 10 REGISTER AREA
0022BD 02BE   0000  A         FDB   TKPCA    STARTING ADDRESS FOR PROCESS 10

00230D 02C0   003B  A         RMB   56       TASK 11 STACK AREA
00231         02FB  D TSKNMB  EQU   *
00232D 02FB   0006  A         RMB   6        TASK 11 REGISTER AREA
00233D 02FE   0000  A         FDB   TKPCB    STARTING ADDRESS FOR PROCESS 11

00235D 0300   003B  A         RMB   56       TASK 12 STACK AREA
00236         033B  D TSKNMC  EQU   *
00237D 033B   0006  A         RMB   6        TASK 12 REGISTER AREA
0023BD 033E   0000  A         FDB   TKPCC    STARTING ADDRESS FOR PROCESS 12

00240D 0340   003B  A         RMB   56       TASK 13 STACK AREA
00241         037B  D TSKNMD  EQU   *
00242D 037B   0006  A         RMB   6        TASK 13 REGISTER AREA
00243D 037E   0000  A         FDB   TKPCD    STARTING ADDRESS FOR PROCESS 13

00245D 0380   003B  A         RMB   56       TASK 14 STACK AREA
00246         03BB  D TSKNAM  EQU   *
00247D 03BB   0006  A         RMB   6        TASK 14 REGISTER AREA
0024BD 03BE   0000  A         FDB   TKPCM    STARTING ADDRESS FOR PROCESS 14



00250                 *  EXTRA VARIABLES

00252                 *  THERE IS NO ROOM FOR THESE VARIABLES IN THE BASE PAGE.
00253                 *  IT IS POSSIBLE THROUGH OPTIMIZATION TO REDUCE THE NUMBER
00254                 *  OF VARIABLES CONTAINED IN THE BASE PAGE.

00256D 03C0   0001  A IRQBIT RMB   1        COUNTS THE NUMBER OF DEFECTIVE ACIA'S
00257D 03C1   0002  A TEMP1  RMB   2        TEMPORARY LOCATION USED BY (TSKCHG)
0025B         03C3  D VARTER EQU   *        END OF VARIABLE LIST
```

```
00260                   *  TABLE: IRQTBL

00262                   *  THIS MONITOR SUPPORTS UP TO FOUR OEVICES.  EACH OEVICE IS
00263                   *  OEFINEO THROUGH A SERIES OF TWELVE BYTES.  A SERVICE AOORESS
00264                   *  OF ZERO WILL OECLARE THAT OEVICE TO BE NONEXISTENT.
00265                   *  TABLE OEFINITIONS USE ONE BYTE POINTERS.  THEY
00266                   *  REFERENCE THE BASE PAGE.
00267                   *  BYTE OEFINITIONS:
0026B                   *    0   INPUT QUEUE STARTING LOCATION
00269                   *    1 - INPUT QUEUE ENOING LOCATION PLUS ONE
00270                   *        OUTPUT QUEUE STARTING LOCATION
00271                   *    2 - OUTPUT QUEUE ENOING LOCATION PLUS ONE
00272                   *    3 - CORRESPONOING RAM POINTER INTO (VARQUE)
00273                   *    4 - SERVICE ROUTINE POINTER FOR OEVICE
00274                   *    6 - OEVICE AOORESS
00275                   *    B - ATTRIBUTE POINTER FOR OEVICE
00276                   *    10 - REQUEST-TO-SENO (RTS) AOORESS PORT POINTING


00278P 0000                       PSCT


00280                   *  OEFINITION OF OEVICE ONE

002B2          0000   P IRQTBL EQU    *
002B3P 0000    00     B NETOTE FCB    QUETBL
00284P 0001    10     B        FCB    QUETBL+BLOCK
00285P 0002    20     B        FCB    QUETBL+(2*BLOCK)
00286P 0003    80     B        FCB    VARQUE
002B7P 0004    0210   P        FOB    ACIASV
002BBP 0006    ECCO   A        FOB    NETWRK
00289P 000B    00E5   B        FOB    PARMPT
00290P 000A    ECOO   A        FOB    RTSOOO
```

```
00292                   *  DEFINITION OF DEVICE TWO

00294P 000C    20    B NETDCE FC8    QUET8L+(2*8LOCK)
00295P 000D    30    B        FC8    QUET8L+(3*8LOCK)
00296P 000E    40    B        FC8    QUET8L+(4*8LOCK)
00297P 000F    8E    B        FC8    VARQUE+14
00298P 0010    0210  P        FD8    ACIASV
00299P 0012    ECC4  A        FD8    SU8NET
00300P 0014    00E7  B        FD8    PARMPT+2
00301P 0016    ECD1  A        FD8    RTSD01


00303                   *  DEFINITION OF DEVICE THREE

00305P 0018    40    B USRUSR FC8    QUET8L+(4*8LOCK)
00306P 0019    50    B        FC8    QUET8L+(5*8LOCK)
00307P 001A    60    B        FC8    QUET8L+(6*8LOCK)
00308P 0018    9C    B        FC8    VARQUE+28
00309P 001C    0210  P        FD8    ACIASV
00310P 001E    FCF4  A        FD8    URACIA
00311P 0020    00E9  B        FD8    PARMPT+4
00312P 0022    0000  A        FD8    RTSD03


00314                   *  DEFINITION OF DEVICE FOUR

00316P 0024    60    B NETUSR FC8    QUET8L+(6*8LOCK)
00317P 0025    70    B        FC8    QUET8L+(7*8LOCK)
00318P 0026    80    B        FC8    QUET8L+(8*8LOCK)
00319P 0027    AA    B        FC8    VARQUE+42
00320P 0028    0210  P        FD8    ACIASV
00321P 002A    ECC8  A        FD8    S8ACIA
00322P 002C    00E8  B        FD8    PARMPT+6
00323P 002E    ECD2  A        FD8    RTSD02


00325                   *  THIS LABEL PERMITS THE ADDITION OF NUMBERS EASILY
00326                   *  TO THE TABLE ADDRESS.  IN ANOTHER WORDS, A DISPLACEMENT
00327                   *  FROM THE BASE ADDRESS.

00329P 0030    0000  P IRQLST FD8    IRQT8L
```

```
00331                   *  TABLE: STRAOR

00333                   *  THIS TABLE IS USED AS THE SOURCE INPUT TO INITIALIZE THE
00334                   *  (STCKBL) TABLE.  THIS TABLE CONTAINS THE TRUE STARTING
00335                   *  LOCATIONS OF EACH OF THE PROCESSES INCLUDING THE
00336                   *  PRIORITY PROCESS.

0033BP 0032    0000  A STRADR FDB    TKPC0
00339P 0034    0000  A        FDB    TKPC1
0034OP 0036    0000  A        FDB    TKPC2
00341P 003B    0000  A        FDB    TKPC3
00342P 003A    0000  A        FDB    TKPC4
00343P 003C    0000  A        FDB    TKPC5
00344P 003E    0000  A        FDB    TKPC6
00345P 0040    0000  A        FOB    TKPC7
00346P 0042    0000  A        FDB    TKPCB
00347P 0044    0000  A        FDB    TKPC9
0034BP 0046    0000  A        FDB    TKPCA
00349P 004B    0000  A        FDB    TKPCB
00350P 004A    0000  A        FDB    TKPCC
00351P 004C    0000  A        FDB    TKPCD
00352P 004E    0000  A        FDB    TKPCM
```

```
00354                   *  TABLE: INTQUE

00356                   *  THIS TABLE IS THE INITIAL (VARQUE) TABLE.  IT IS TRANSFERRED
00357                   *  INTO RAM BY THE INITIALIZATION ROUTINE.  (VARQUE) IS LOCATED
0035B                   *  IN THE BASE PAGE.  (VARQUE) DEFINES THE DEVICE QUEUES.  EACH
00359                   *  DEFINITION REQUIRES SEVEN BYTES WITH TWO QUEUES PER DEVICE.
00360                   *  THE POINTER DEFINITIONS ARE SIXTEEN BITS IN LENGTH EVEN
00361                   *  THOUGH THE BASE PAGE IS BEING REFERENCED.
00362                   *  BYTE DEFINITION:
00363                   *  0  - INPUT QUEUE FRONT POINTER
00364                   *  2  - INPUT QUEUE REAR POINTER
00365                   *  4  - HOLDING AREA FOR BLOCKED TASK
00366                   *  6  - INPUT QUEUE COUNTER
00367                   *  7  - OUTPUT QUEUE FRONT POINTER
0036B                   *  9  - OUTPUT QUEUE REAR POINTER
00369                   *  11   HOLDING AREA FOR BLOCKED TASK
00370                   *  13   OUTPUT QUEUE COUNTER



00372                   *  DEFINITIION FOR DEVICE ONE INPUT QUEUE

00374          0050  P INTQUE EQU    *
00375P 0050    0000  B        FDB    QUETBL
00376P 0052    0000  B        FDB    QUETBL
00377P 0054    0000  A        FDB    $0000
0037BP 0056    00    A        FCB    $00


003B0                    *  DEFINITION FOR DEVICE ONE OUTPUT QUEUE

003B2P 0057    0010  B        FDB    QUETBL+BLOCK
003B3P 0059    0010  B        FDB    QUETBL+BLOCK
003B4P 005B    0000  A        FDB    $0000
003B5P 0050    00    A        FCB    $00
```

```
003B7                   *  DEFINITION FOR DEVICE TWO INPUT QUEUE

003B9P 005E   0020  B       FDB    QUETBL+(2*BLOCK)
00390P 0060   0020  B       FDB    QUETBL+(2*BLOCK)
00391P 0062   0000  A       FDB    $0000
00392P 0064   00    A       FCB    $00


00394                   *  DEFINITION FOR DEVICE TWO OUTPUT QUEUE

00396P 0065   0030  B       FDB    QUETBL+(3*BLOCK)
00397P 0067   0030  B       FDB    QUETBL+(3*BLOCK)
0039BP 0069   0000  A       FDB    $0000
00399P 006B   00    A       FCB    $00


00401                   *  DEFINITION FOR DEVICE THREE INPUT QUEUE

00403P 006C   0040  B       FDB    QUETBL+(4*BLOCK)
00404P 006E   0040  B       FDB    QUETBL+(4*BLOCK)
00405P 0070   0000  A       FDB    $0000
00406P 0072   00    A       FCB    $00


0040B                   *  DEFINITION FOR DEVICE THREE OUTPUT QUEUE

00410P 0073   0050  B       FDB    QUETBL+(5*BLOCK)
00411P 0075   0050  B       FDB    QUETBL+(5*BLOCK)
00412P 0077   0000  A       FDB    $0000
00413P 0079   00    A       FCB    $00


00415                   *  DEFINITION FOR DEVICE FOUR INPUT QUEUE

00417P 007A   0060  B       FDB    QUETBL+(6*BLOCK)
0041BP 007C   0060  B       FDB    QUETBL+(6*BLOCK)
00419P 007E   0000  A       FDB    $0000
00420P 00B0   00    A       FCB    $00


00422                   *  DEFINITION FOR DEVICE FOUR OUTPUT QUEUE

00424P 00B1   0070  B       FDB    QUETBL+(7*BLOCK)
00425P 00B3   0070  B       FDB    QUETBL+(7*BLOCK)
00426P 00B5   0000  A       FDB    $0000
00427P 00B7   00    A       FCB    $00
```

```
00429                   *  TABLE: RESTSK

00431                   *  THIS IS THE STATIC STACK POINTER AND STATUS TABLE.  (PROCSS)
00432                   *  IS INITIALIZED BY THIS TABLE THROUGH THE INITIALIZATION ROUTINE.
00433                   *  THERE ARE FOURTEEN PROCESSES PLUS ONE PRIORITY PROCESS.
00434                   *  NOTE:  PROCESS 6 AND D ARE BLOCKED.  THEY ARE READIED THROUGH
00435                   *         SYSTEM CONTROL.  THE PRIORITY PROCESS IS NOT USED.
00436                   *  STATUS DEFINITIONS:
00437                   *  0  - TASK IS READIED
00438                   *  <0 - TASK IS PERMANENTLY BLOCKED
00439                   *  >0 - TASK IS TEMPERARILY BLOCKED FOR THE AMOUNT OF TIME
00440                   *       SPECIFIED - TIME IS DEFINED IN PRIMARY TICKS


00442          0088  P RESTSK EQU    *

00444P 0088    0038  D        FDB    TSKNM0
00445P 008A    00    A        FCB    $00

00447P 0088    0078  0        FDB    TSKNM1
00448P 0080    00    A        FCB    $00

00450P 008E    0088  D        FDB    TSKNM2
00451P 0090    00    A        FCB    $00
```

```
00453P 0091    00F8  D        FDB    TSKNM3
00454P 0093    00    A        FCB    $00

00456P 0094    0138  D        FDB    TSKNM4
00457P 0096    00    A        FCB    $00

00459P 0097    0178  D        FDB    TSKNM5
00460P 0099    00    A        FCB    $00

00462P 009A    01B8  D        FDB    TSKNM6
00463P 009C    80    A        FCB    $80

00465P 009D    01F8  D        FDB    TSKNM7
00466P 009F    00    A        FCB    $00

00468P 00A0    0238  D        FDB    TSKNM8
00469P 00A2    00    A        FCB    $00

00471P 00A3    0278  D        FDB    TSKNM9
00472P 00A5    00    A        FCB    $00

00474P 00A6    02B8  D        FDB    TSKNMA
00475P 00A8    00    A        FCB    $00

00477P 00A9    02F8  D        FDB    TSKNMB
00478P 00AB    00    A        FCB    $00

00480P 00AC    0338  D        FDB    TSKNMC
00481P 00AE    00    A        FCB    $00

00483P 00AF    0378  D        FDB    TSKNMD
00484P 00B1    80    A        FCB    $80

00486P 00B2    03B8  D RESPRI FDB    TSKNAM
00487P 00B4    80    A        FCB    $80
```

```
                        *  TABLE: PARMT

                        *  THIS IS THE DEVICE ATTRIBUTE TABLE.  IT IS USED AS A DEDICATED
                        *  HOLDING AREA FOR EACH DEVICE.  PRESENTLY, IT IS USED BY THE
                        *  ACIA SERVICE ROUTINE.  THIS TABLE IS TRANSFERRED INTO THE
                        *  (PARMPT) TABLE DURING SYSTEM INITIALIZATION.
                        *  BYTE DEFINITIONS:
                        *  BYTE 0 -
                        *       MSB - SET TO ENABLE TRANSMITTING INTERRUPT
                        *       LSB - SET TO DISABLE REQUEST-TO-SEND (RTS) LINE
                        *  BYTE 1 - SET IF DEVICE IS NOT FUNCTIONING PROPERLY


                  00B5  P PARMT  EQU    *

                  00    A        FCB    00,00    ATTRIBUTES FOR DEVICE ONE
                  00    A
                  00    A        FCB    00,00    ATTRIBUTES FOR DEVICE TWO
                  00    A
                  00    A        FCB    00,00    ATTRIBUTES FOR DEVICE THREE
                  00    A
                  00    A        FCB    00,00    ATTRIBUTES FOR DEVICE FOUR
                  00    A


                        *  TABLE: TSKTBL

                        *  THIS IS AN INITIALIZATION TABLE.  IT IS TRANSFERRED INTO THE
                        *  BASE PAGE.  THE FOLLOWING PARAMETERS ARE INITIALIZED:
                        *  TSKPTR, IRQREC, IRQVAL, QUEREC, DATA, CRTFLG, SUBVAL, AND PRIVAL.
                        *  THE CONSTANTS (SUBTIC) AND (PRITIC) ARE USED IN THE (NMISVC)
                        *  ROUTINE TO RE-INITIALIZE (SUBVAL) AND (PRIVAL).


                  00BD  P TSKTBL EQU    *

                  00B8  B        FDB    PROCSS   INITIALIZE (TSKPTR)
                  0000  P        FDB    IRQTBL   INITIALIZE (IRQREC)
                  0000  P        FDB    IRQTBL   INITIALIZE (IRQVAL)
                  0080  B        FDB    VARQUE   INITIALIZE (QUEREC)
                  00    A        FCB    $00      INITIALIZE (DATA)
                  00    A        FCB    $00      INITIALIZE (CRTFLG)
                  02    A SUBTIC FCB    $02      INITIALIZE (SUBVAL)
                  02    A PRITIC FCB    $02      INITIALIZE (PRIVAL)
```

```
00527                   ************************************************************************
00528                   *                                                                      *
00529                   *    MCNITOR INITIALIZATION                                            *
00530                   *                                                                      *
00531                   *    THIS ROUTINE INITIALIZES RAM.  THERE ARE EIGHT ASSIGNEO           *
00532                   *    FUNCTIONS TO THIS ROUTINE.  THEY ARE:                             *
00533                   *    1)  SET THE STACK POINTER                                         *
00534                   *    2)  DISABLE NMI FOR THIS ROUTINE                                  *
00535                   *    3)  SET INTERRUPT VECTORS                                         *
00536                   *    4)  CLEAR STACK AREA                                              *
00537                   *    5)  INITIALIZE OTE AND OCE                                        *
00538                   *    6)  INITIALIZE OEVICES                                            *
00539                   *    7)  INITIALIZE STACK AREA                                         *
00540                   *    8)  TRANSFER INITIALIZATION TA8LES                                *
00541                   *                                                                      *
00542                   ************************************************************************



00544                   *  THE STACK POINTER IS INITIALIZED TO POINT TO THE FIRST TASK
00545                   *  STACK AREA.  THIS IS AN AR8ITRARY TASK SELECTION.  IF AN
00546                   *  INTERRUPT OCCURS 8EFORE THIS POINT, THE SYSTEM WILL 8OMB.

00548           00C9  P RSTSVC EQU    *
00549P 00C9 8E 0038   0        LOS    #TSKNMO


00551                   *  IT IS ASSUMEO THAT AN NMI MAY OCCUR.  IF IT OOES, THE (NMISVC)
00552                   *  ROUTINE IS NOT READY TO RECEIVE AN INTERRUPT.  THEREFORE,
00553                   *  THE (NMISVC) ROUTINE WILL IGNORE THE INTERRUPT.
00554                   *  SETTING THE (HOLDIT) FLAG WILL ACCOMPLISH THIS.

00556P 00CC 86 01     A        LDAA   #1
00557P 00CE 97 FE     B        STAA   HOLDIT



00559                   *  SET INTERRUPT VECTORS

00561                   *  THE IRQSVC, SWISVC, AND NMISVC INTERRUPT VECTORS ARE STORED
00562                   *  IN LOCATIONS $FFF8, $FFFA, ANO $FFFC, RESPECTIVELY.
00563                   *  IT IS ASSUMED THAT THIS ROUTINE WILL COMPLETE 8EFORE
00564                   *  AN INTERRUPT OCCURS.

00566           00D0  P LDSVCS EQU    *
00567P 00D0 CE 018C   P        LDX    #IRQSVC
00568P 00D3 FF FFF8   A        STX    $FFF8    IRQ VECTOR
00569P 00D6 CE 0429   P        LDX    #SWISVC
00570P 00D9 FF FFFA   A        STX    $FFFA    SWI VECTOR
00571P 00DC CE 0433   P        LDX    #NMISVC
00572P 00DF FF FFFC   A        STX    $FFFC    NMI VECTOR
```

```
00574                   *  CLEAR STACK AREA

00576                   *  THE ENTIRE STACK AREA IS CLEARED.  ZERO'ING BEGINS AT (STCKBL)
00577                   *  AND CONTINUES UP TO (VARTER).  THE ADDITIONAL VARIABLES ARE
00578                   *  ALSO CLEARED.  IT SHOULD BE NOTED THAT THE REGISTER AREAS WILL
00579                   *  BE INITIALIZED TO ZERO AND SO WILL THE CONDITIONAL CODE
00580                   *  REGISTER.  THIS ACTION ENABLES THE INTERRUPT MASK OF THE
00581                   *  PROCESSOR.

00583P 00E2 CE 0000  0          LDX    #STCKBL
00584P 00E5 6F 00    A MORCLR   CLR    0,X
00585P 00E7 08                  INX
00586P 00E8 BC 03C3  0          CPX    #VARTER
00587P 00EB 26 FB 00E5          BNE    MORCLR



00589                   *  INITIALIZE OTE AND OCE

00591                   *  THIS MONITOR HAS THE CAPABILITY OF INITIALIZING EXTERNAL
00592                   *  ROUTINES.  THIS FEATURE AVOIDS HAVING A SEPERATE PROCESS
00593                   *  WASTED FOR INITIALIZATION.

00595P 00E0 BD 0000  A          JSR    INTOTE   INITIALIZATION FOR THE OTE
00596P 00F0 BD 0000  A          JSR    INTOCE   INITIALIZATION FOR THE OCE
```

```
00598                   *  INITIALIZE DEVICES

00600                   *  THIS ROUTINE IS SUBDIVIDED INTO THREE SECTIONS;
00601                   *  REQUEST-TO-SEND (RTS), ACIA, AND PIA INITIALIZATIONS.

00603                   *  THE REQUEST-TO-SEND (RTS) LINE IS ENABLED FOR ALL DEVICES.
00604                   *  THIS IS ACCOMPLISHED THROUGH THE (RTSDD) PORTS.

00606           00F3  P INTDEV EQU    *
00607P 00F3 4F                 CLRA
00608P 00F4 C6 04     A        LDAB   #DEVICE
00609P 00F6 CE ECD0   A        LDX    #RTSDDD
00610P 00F9 A7 00     A ITDVLP STAA   0,X
00611P 00FB 08                 INX
00612P 00FC 5A                 DECB
00613P 00FD 26 FA 00F9         BNE    ITDVLP


00615                   *  ACIA INITIALIZATION.  THE CONTROL WORD IS (ACNTRL).
00616                   *  CONTROL WORD OF $D5 SPECIFIES A BAUD RATE OF SIXTEEN TIMES,
00617                   *  A TRANSMITTED OR RECEIVED WORD CONSISTING OF EIGHT BITS,
00618                   *  THE TRANSMITTING INTERRUPT DISABLED, AND THE RECIEVING
00619                   *  INTERRUPT ENABLED.

00621P 00FF C6 03     A        LDAB   #$03     RESET VALUE
00622P 0101 B6 D5     A        LDAA   #ACNTRL  INITIALIZATION VALUE
00623P 0103 F7 ECC0   A        STAB   NETWRK   NETWORK PORT DTE TO DCE
00624P 0106 B7 ECC0   A        STAA   NETWRK

00626P 0109 F7 ECC4   A        STAB   SUBNET   NETWORK PORT DCE TO DTE
00627P 010C B7 ECC4   A        STAA   SUBNET

00629P 010F F7 FCF4   A        STAB   URACIA   USER PORT TO DTE
00630P 0112 B7 FCF4   A        STAA   URACIA

00632P 0115 F7 ECC8   A        STAB   SBACIA   NETWORK PORT TO DCE
00633P 0118 B7 ECC8   A        STAA   SBACIA

00635P 011B F7 ECCC   A        STAB   $ECCC    CLEAR FIFTH ACIA.  NOT USED HERE.


00637                   *  PIA INITIALIZATION.
00638                   *  SIDE A IS ASSIGNED AS INPUT.  CA1 ACTIVATED LOW INDICATING THAT
00639                   *  THERE IS VALID DATA.  CA2 IS ACTIVATED BY THE MONITOR SIGNIFING
00640                   *  ACCEPTANCE OF DATA.
00641                   *  SIDE B IS ASSIGNED AS OUTPUT.  CB2 IS ACTIVE LOW INDICATING THE
00642                   *  PRESENCE OF VALID DATA.  CB1 IS ACTIVE LOW INDICATING ACCEPTANCE
00643                   *  OF VALID DATA BY PERIPHERAL.

00645P 011E B6 20     A        LDAA   #$20
00646P 0120 B7 9001   A        STAA   USRPIA+1
00647P 0123 B6 FF     A        LDAA   #$FF
00648P 0125 B7 9003   A        STAA   USRPIA+3
00649P 0128 B6 25     A        LDAA   #$25
00650P 012A B7 9003   A        STAA   USRPIA+3
```

```
00652                    *  INITIALIZE STACK AREA

00654                    *  THIS ROUTINE INITIALIZES THE PROGRAM COUNTERS LOCATED IN THE
00655                    *  STACK AREA OF (STCKBL).  THE SOURCE FOR THE INITIALIZATION
00656                    *  IS TABLE (STRAOR).

00658           0120  P LOAOR  EQU    *
00659P 0120 C6 0F     A        LOAB   #NTASK+1
00660P 012F 07 F0     B        STAB   COUNT    INITIALIZE COUNTER FOR LOOPPING
00661P 0131 CE 0032   P        LOX    #STRAOR
00662P 0134 0F F9     B        STX    SOURC    INITIALIZE SOURCE POINTER
00663P 0136 CE 0038   0        LOX    #TSKNMO
00664P 0139 0F FB     B        STX    DESTIN   INITIALIZE DESTINATION POINTER

00666                    *  RETRIEVE POINTER FROM SOURCE

00668P 013B 6F 01     A AORLP1 CLR    1,X
00669P 013D 0E F9     B        LOX    SOURC
00670P 013F A6 00     A        LOAA   0,X
00671P 0141 E6 01     A        LOAB   1,X
00672P 0143 08                 INX
00673P 0144 08                 INX
00674P 0145 0F F9     B        STX    SOURC

00676                    *  STORE POINTER AT DESTINATION ADDRESS AND INCREMENT ADDRESS

00678P 0147 0E F8     B        LOX    DESTIN
00679P 0149 A7 06     A        STAA   6,X
00680P 014B E7 07     A        STAB   7,X
00681P 014D 86 40     A        LOAA   #64
00682P 014F 9B FC     B        ADDA   DESTIN+1
00683P 0151 97 FC     B        STAA   DESTIN+1
00684P 0153 24 03 0158         BCC    *+5
00685P 0155 7C 00F8   B        INC    DESTIN
00686P 0158 7A 00F0   B        DEC    COUNT    DECREMENT COUNTER
00687P 015B 26 0E 0138         BNE    AORLP1   REPEAT THE LOOP IF NECESSARY
```

```
006B9                    *  TRANSFER INITIALIZATION TABLES

00691                    *  THE INITIALIZATION TABLES (INTQUE), (RESTSK), (PARMT) AND
00692                    *  (TSKTBL) ARE TRANSFERRED TO THE BASE PAGE.  THIS PROCEDURE
00693                    *  SETS UP THE PAGE FOR MULTI-TASKING OPERATONS.

00695          015D  P INTRAM EQU    *
00696P 015D CE 0050  P         LOX    #INTQUE
00697P 0160 DF F9    B         STX    SOURC     INITIALIZE SOURCE POINTER
0069BP 0162 CE 00B0  B         LDX    #VARQUE
00699P 0165 DF FB    B         STX    OESTIN    INITIALIZE OESTINATION POINTER

00701                    *  FETCH DATA FROM SOURCE AODRESS

00703P 0167 DE F9    B RSTLP1 LDX    SOURC
00704P 0169 A6 00    A         LDAA   0,X
00705P 016B 0B                 INX
00706P 016C DF F9    B         STX    SOURC

00708                    *  STORE DATA AT OESTINATION AODRESS

00710P 016E DE FB    B         LDX    OESTIN
00711P 0170 A7 00    A         STAA   0,X
00712P 0172 0B                 INX
00713P 0173 DF FB    B         STX    OESTIN
00714P 0175 BC 00F9  B         CPX    #SOURC    CHECK FOR POSSIBLE TERMINATION
00715P 0178 26 ED 0167         BNE    RSTLP1    REPEAT IF NECESSARY


00717                    *  CLEAR REMAINDER OF BASE PAGE

00719P 017A 4F                 CLRA
00720P 017B A7 00    A RSTLP2 STAA   0,X
00721P 017D 0B                 INX
00722P 017E BC 00FD  B         CPX    #INTEND   CHECK FOR TERMINATION OF SE PAGE
00723P 0181 26 FB 017B         BNE    RSTLP2    REPEAT IF NECESSARY



00725                    *  ENABLE ACCESS TO SCHEOULER

00727P 0183 4A                 DECA
00728P 0184 97 F9    B         STAA   INTFLG



00730                    *  ENABLE NMI SERVICE ROUTINE AND ENTER SCHEDULER

00732P 0186 7F 00FE  B         CLR    HOLDIT
00733P 0189 7E 047B  P         JMP    TSKCHG
```

```
DD735                   ************************************************************************
DD736                   *                                                                      *
DD737                   *    IRQ INTERRUPT SERVICE RDUTINE                                     *
DD738                   *                                                                      *
DD739                   *    IRQ INTERRUPTS DRIGINATE FRDM THE I/D DEVICES WHICH REQUIRE       *
DD74D                   *    IMMEDIATE ATTENTIDN.  EACH DEVICE CDULD HAVE UP TD TWD            *
DD741                   *    PDSSIBLE INTERRUPTS.  THE RDUTINE WILL SEQUENCE THRDUGH THE       *
DD742                   *    (IRQTBL) TABLE FDR EACH INTERRUPT.  FDR EACH DEVICE, THE          *
DD743                   *    SERVICE RDUTINE IS ENTERED.  THE ENTIRE TABLE IS SCANNED.         *
DD744                   *                                                                      *
DD745                   ************************************************************************


DD747          D18C  P IRQSVC EQU    *
DD748P D18C BD D19D  P         JSR    LDDKUP
DD749P D18F 3B                 RTI



DD751                   *  LDDKUP RDUTINE

DD753                   *  THIS RDUTINE SCANS THE ENTIRE TABLE BEFDRE RETURNING.  IF THE
DD754                   *  ADDRESS DF THE SERVICE RDUTINE IS ZERD THEN THAT SPECIFIC
DD755                   *  DEVICE IS SKIPPED.

DD757          D19D  P LDDKUP EQU    *
DD758P D19D 7C DDF6  B         INC    CRTFLG   SET CRITICAL REGIDN
DD759P D193 B6 DD31  P         LDAA   IRQLST+1
DD76DP D196 97 FD    B         STAA   IRQREC+1 INITIALIZE (IRQREC)
DD761P D198 C6 D4    A         LDAB   #DEVICE
DD762P D19A D7 FD    B         STAB   CDUNT    INITIALIZE CDUNTER

DD764P D19C DE EF    B ICYCLE LDX    IRQREC
DD765P D19E EE D4    A         LDX    4,X      FETCH ADDRESS DF SERVICE RDUTINE
DD766P D1AD 27 D2 D1A4         BEQ    *+4      SKIP ENTRY PDINT IF ADDRESS IS ZERD
DD767P D1A2 AD DB    A         JSR    B,X      ENTRY PDINT FDR SERVICE RDUTINE
DD768P D1A4 7A DDFD  B         DEC    CDUNT    DECREMENT CDUNTER
DD769P D1A7 26 D4 D1AD         BNE    *+6      TEST FDR PDSSIBLE TERMINATIDN
DD77DP D1A9 7A DDF6  B         DEC    CRTFLG   EXIT CRITICAL REGIDN AND RETURN
DD771P D1AC 39                 RTS

DD773                   *  ADVANCE TABLE PDINTER TD NEXT DEVICE

DD775P D1AD B6 DC    A         LDAA   #12
DD776P D1AF 9B FD    B         ADDA   IRQREC+1
DD777P D1B1 97 FD    B         STAA   IRQREC+1
DD778P D1B3 2D E7 D19C         BRA    ICYCLE   REPEAT LDDP
```

```
00780                   ************************************************************************
00781                   *                                                                      *
00782                   *    PIA INTERRUPT SERVICE ROUTINE                                     *
00783                   *                                                                      *
00784                   *    THIS ROUTINE IS CALLED TO SERVICE A PIA FROM THE ROUTINE          *
00785                   *    (LOOKUP).  SIDE A HAS BEEN DEFINED AS THE INPUT AND SIDE          *
00786                   *    B IS DEFINED AS THE OUTPUT.  BOTH SIDES USE FULL HANDSHAKING.     *
00787                   *    THERE ARE FOUR ADDITIONAL ROUTINES WHICH CONTROL THE INTERRUPTS.  *
00788                   *    THESE ROUTINES ARE ACCESSED THROUGH THE FOUR VECTORS PRECEDING    *
00789                   *    THE START OF THIS ROUTINE.  THESE ROUTINES ARE CALL FROM OTHER    *
00790                   *    ROUTINES ELSEWHERE IN THE MONITOR.                                *
00791                   *                                                                      *
00792                   ************************************************************************



00794          0185  P PIASVC EQU    *
00795P 0185    01E0  P        FDB    PAENIN   ENABLE INPUT INTERRUPT VECTOR
00796P 0187    01E6  P        FDB    PADSIN   DISABLE INPUT INTERRUPT VECTOR
00797P 0189    01EC  P        FDB    PAENOT   ENABLE OUTPUT INTERRUPT VECTOR
00798P 0188    01F2  P        FDB    PADSOT   DISABLE OUTPUT INTERRUPT VECTOR

00800                   *  ENTRY POINT FOR SERVICE ROUTINE.

00802P 0180 0E EF    B         LDX    IRQREC
00803P 018F EE 08    A         LDX    8,X
00804P 01C1 A6 01    A         LDAA   1,X      FETCH STATUS WORD FOR SIDE A
00805P 01C3 28 05 01CA         BMI    PAREAD   INTERRUPT PENDING IF VALUE IS NEGATIVE
00806P 01C5 A6 03    A         LDAA   3,X      FETCH STATUS WORK FOR SIDE B
00807P 01C7 28 09 0102         BMI    PIAWRT   INTERRUPT PENDING IF VALUE IS NEGATIVE
00808P 01C9 39                 RTS


00810                   *  SERVICE ROUTINE FOR SIDE A.

00812P 01CA E6 00    A PAREAD LDAB   0,X      REMOVE DATA FROM DEVICE AND CLEAR INTERRUPT
00813P 01CC 07 F5    B         STAB   DATA
00814P 01CE BD 0206  P         JSR    XFERIN   QUEUE DATA FOR INPUT
00815P 0101 39                 RTS


00817                   *  SERVICE ROUTINE FOR SIDE B.

00819P 0102 BD 0304  P PIAWRT JSR    XFEROT   DEQUEUE DATA FOR TRANSMISSION
00820P 0105 D6 F5    B         LDAB   DATA
00821P 0107 0E EF    B         LDX    IRQREC
00822P 0109 EE 06    A         LDX    6,X      FETCH ADDRESS OF OUTPUT PORT
00823P 0108 A6 02    A         LDAA   2,X      CLEAR PENDING INTERRUPT ONLY
00824P 0100 E7 02    A         STAB   2,X      ISSUE DATA TO PORT
00825P 010F 39                 RTS
```

```
00827                    *  ENA8LE INPUT INTERRUPT

00829           01E0  P PAENIN EQU    *
00830P 01E0 86 01     A        LDAA   #$1      ENA8LING WORD
00831P 01E2 80 01F8   P        JSR    PARDIN   CALL INPUT INTERRUPT CONTROL ROUTINE
00832P 01E5 39                 RTS



00834                    *  OISA8LE INPUT INTERRUPT

00836           01E6  P PADSIN EQU    *
00837P 01E6 86 00     A        LDAA   #$0      OISA8LING WORD
00838P 01E8 8D 01F8   P        JSR    PARDIN   CALL INPUT INTERRUPT CONTROL ROUTINE
00839P 01E8 39                 RTS



00841                    *  ENA8LE OUTPUT INTERRUPT

00843           01EC  P PAENOT EQU    *
00844P 01EC 86 01     A        LDAA   #$1      ENA8LING WORD
00845P 01EE 8D 0204   P        JSR    PAWTOT   CALL OUTPUT INTERRUPT CONTROL ROUTINE
00846P 01F1 39                 RTS



00848                    *  DISA8LE OUTPUT INTERRUPT

00850           01F2  P PADSOT EQU    *
00851P 01F2 86 00     A        LDAA   #$0      DISA8LING WORD
00852P 01F4 8D 0204   P        JSR    PAWTOT   CALL OUTPUT INTERRUPT CONTROL ROUTINE
00853P 01F7 39                 RTS
```

```
00855                   *  THE INTERRUPT CONTROL ROUTINE FOR BOTH SIDE A AND SIDE B
00856                   *  IS DEFINED THROUGH THIS MACRO.  THE LOGIC IS TO FETCH THE
00857                   *  CONTROL WORD, SET OR CLEAR THE LEAST SIGNIFICANT BIT
00858                   *  AND REPLACE IT.  THE DIFFERENCE BETWEEN THE TWO ROUTINES
00859                   *  IS THE REGISTER THAT IS USED.

00861                   PACNTL MACR   C
00862                     LDX IRQREC
00863                     LDX 6,X  FETCH ADDRESS OF DEVICE
00864                     LDA B 0,X  FETCH CONTROL WORD FROM APPROPRIATE REGISTER
00865                     AND A #$FE
00866                     ABA  SET OR CLEAR LEAST SIGNIFICANT BIT
00867                     STA A 0,X  REPLACE UPDATED CONTROL WORD
00868                     ENDM



00870                   *  THIS ROUTINE WILL MODIFY THE REGISTER ASSOCIATED WITH SIDE A.

00872          01F8  P PARDIN EQU    *
00873P 01F8                   PACNTL 1        MACRO CALL FOR SIDE A
     P 01F8 DE EF     B       LDX    IRQREC
     P 01FA EE 06     A       LDX    6,X      FETCH ADDRESS OF DEVICE
     P 01FC E6 01     A       LDAB   1,X      FETCH CONTROL WORD FROM APPROPRIATE REGISTER
     P 01FE 84 FE     A       ANDA   #$FE
     P 0200 18               ABA            SET OR CLEAR LEAST SIGNIFICANT BIT
     P 0201 A7 01     A       STAA   1,X      REPLACE UPOATEO CONTROL WORD
00874P 0203 39               RTS



00876                   *  ROUTINE FOR MODIFING REGISTER ASSOCIATED WITH SIDE B

00878          0204  P PAWTOT EQU    *
00879P 0204                   PACNTL 3        MACRO CALL FOR MODIFING SIDE B REGISTER
     P 0204 DE EF     B       LDX    IRQREC
     P 0206 EE 06     A       LDX    6,X      FETCH ADDRESS OF DEVICE
     P 0208 E6 03     A       LDAB   3,X      FETCH CONTROL WORD FROM APPROPRIATE REGISTER
     P 020A 84 FE     A       ANDA   #$FE
     P 020C 18               ABA            SET OR CLEAR LEAST SIGNIFICANT BIT
     P 020D A7 03     A       STAA   3,X      REPLACE UPDATED CONTROL WORD
00880P 020F 39               RTS
```

```
00882                  ************************************************************************
00883                  *                                                                      *
00884                  *    ACIA INTERRUPT SERVICE ROUTINE                                    *
00885                  *                                                                      *
00886                  *    THIS ROUTINE IS CALLED TO SERVICE AN ACIA FROM THE ROUTINE        *
00887                  *    (LOOKUP).  THERE ARE FOUR ROUTINES WHICH CONTROL THE INTERRUPTS   *
00888                  *    OF THE ACIA.  THESE ROUTINES ARE ACCESSED THROUGH THE VECTORS     *
00889                  *    WHICH PRECEDE THE START OF THIS ROUTINE.  SPECIAL CONSIDERATION   *
00890                  *    IS MADE IF AN OVERRUN OR FRAME ERROR IS DETECTED.  THE SERVICE    *
00891                  *    ROUTINE WILL DETECT A FAULTY TRANSMISSION OR RECEPTION ERROR      *
00892                  *    AND ACT ACCORDINGLY.                                              *
00893                  *                                                                      *
00894                  ************************************************************************


00896           0210  P ACIASV EQU    *
00897P 0210     0291  P        FDB    ACENIN   ENABLE INPUT INTERRUPT VECTOR
00898P 0212     0295  P        FDB    ACDSIN   DISABLE INPUT INTERRUPT VECTOR
00899P 0214     02AC  P        FDB    ACENOT   ENABLE OUTPUT INTERRUPT VECTOR
00900P 0216     0283  P        FDB    ACDSOT   DISABLE OUTPUT INTERRUPT VECTOR
00901P 0218 DE EF     B        LDX    IRQREC
00902P 021A EE 06     A        LDX    6,X      FETCH DEVICE ADDRESS
00903P 021C A6 00     A        LDAA   0,X      FETCH STATUS WORD
00904P 021E 2A 29  0249        BPL    IRQCHK   PENDING INTERRUPT IF VALUE IS NEGATIVE


00906                  *  AN INTERRUPT HAS OCCURRED.
00907                  *  CHECK THE ATTRIBUTE TABLE TO DETERMINE WHETHER THIS DEVICE
00908                  *  ALREADY HAS A PENDING INTERRUPT.  IF ONE EXISTS, SKIP THE
00909                  *  ERROR CHECKING ROUTINE WHICH IMMEDIATELY FOLLOWS.

00911P 0220 DE EF     B        LDX    IRQREC
00912P 0222 EE 08     A        LDX    8,X      FETCH THE ATTRIBUTE TABLE POINTER
00913P 0224 6D 01     A        TST    1,X      DETERMINE STATE OF THE DEVICE
00914P 0226 26 05  022D        BNE    XMITB    VALUE IS ZERO FOR A NORMAL DEVICE
00915P 0228 8D 0281  P         JSR    ACHECK   CALL ERROR CHECKING ROUTINE
00916P 0228 25 48  0278        BCS    ACIAER   CARRY IS SET IF AN ERROR EXITS
00917P 022D 85 01     A XMITB  BITA   #01      DETERMINE PENDING INTERRUPT FROM TRANSMITTER
00918P 022F 26 0C  023D        BNE    RCDATA   VALUE IS ZERO FOR TRANSMITTING INTERRUPT


00920                  *  INTERRUPT SERVICE ROUTINE FOR TRANSMITTER

00922P 0231 8D 0304  P         JSR    XFEROT   DEQUEUE DATA TO BE TRANSMITTED
00923P 0234 DE EF     B        LDX    IRQREC
00924P 0236 EE 06     A        LDX    6,X      FETCH DEVICE ADDRESS
00925P 0238 D6 F5     B        LDAB   DATA
00926P 023A E7 01     A        STAB   1,X      ISSUE DATA TO TRANSMITTING PORT
00927P 023C 39                 RTS
```

```
00929                    *  INTERRUPT SERVICE ROUTINE FOR RECEIVER

00931P 023D DE EF     B RCOATA LDX    IRQREC
00932P 023F EE 06     A        LDX    6,X      FETCH DEVICE ADDRESS
00933P 0241 E6 01     A        LDAB   1,X      FETCH RECEIVED DATA FROM DEVICE PORT
00934P 0243 D7 F5     B        STAB   DATA
00935P 0245 BD 02D6   P        JSR    XFERIN   QUEUE DATA AT INPUT QUEUE
00936P 024B 39                 RTS



00938                    *  THIS ROUTINE IS ENTERED IF THE DEVICE DOES NOT SHOW A PENDING
00939                    *  INTERRUPT FROM ITS STATUS REGISTER.  A FURTHER CHECK IS MADE
00940                    *  AT THE ATTRIBUTE TABLE.  IF THE VALUE SHOWS A VALID DEVICE,
00941                    *  THE ROUTINE EXITS, ELSE, A CHECK FOR THE ERROR IS MADE AGAIN.

00943P 0249 DE EF     B IRQCHK LDX    IRQREC
00944P 024B EE 0B     A        LDX    B,X      FETCH ATTRIBUTE TABLE ADDRESS
00945P 024D E6 01     A        LDAB   1,X      FETCH ATTRIBUTE
00946P 024F 27 1F 0270         BEQ    EXIT4    VALID DEVICE IF VALUE IS ZERO
00947P 0251 BD 02B1   P        JSR    ACHECK   CONFIRM THE ERROR EXISTANCE
0094BP 0254 25 1B 0271         BCS    EXIT3    CARRY BIT IS SET IF ERROR EXITS


00950                    *  PENDING INTERRUPT DOES NOT EXIST ANY MORE.  ENTER THIS
00951                    *  ROUTINE AND RESTORE DEVICE TO NORMAL STATE.  THERE ARE
00952                    *  THREE FUNCTIONS OF THIS ROUTINE:
00953                    *  1)  DECREMENT (IRQBIT) [ERROR DEVICE COUNTER]
00954                    *  2)  CLEAR ATTRIBUTE TABLE [WORD ONE]
00955                    *  3)  RESTORE INTERRUPT CONTROL TO DEVICE

00957P 0256 7A 03C0   D        DEC    IRQBIT   DECREMENT ERROR'EO DEVICE COUNTER
0095BP 0259 6F 01     A        CLR    1,X      CLEAR ATTRIBUTE ERROR INDICATOR
00959P 025B B6 B0     A        LDAA   #$B0
00960P 025D BA D5     A ACIASR ORAA   #ACNTRL
00961P 025F B4 9F     A        ANDA   #$9F     FORM CONTROL WORD WITH XMIT INTERRUPT

00963                    *  DETERMINE PREVIOUS STATUS OF XMITTER INTERRUPT

00965P 0261 C6 20     A        LDAB   #$20     TRANSMITTER WORD ENABLE
00966P 0263 6D 00     A        TST    0,X      CONSULT ATTRIBUTE WORD ZERO FOR PREVIOUS STATUS
00967P 0265 2B 00 0267         BMI    *+2      TRANSMITTER ENABLED IF VALUE NEGITIVE
0096BP 0267 C6 40     A        LDAB   #$40     TRANSMITTER WORD DISABLE
00969P 0269 1B                 ABA             SUPERIMPOSE TRANSMITTER WORD WITH CONTROL WORD
00970P 026A DE EF     B        LDX    IRQREC
00971P 026C EE 06     A        LDX    6,X      FETCH DEVICE ADDRESS
00972P 026E A7 00     A        STAA   0,X      RESTORE CONTROL WORD WITH PROPER ATTRIBUTES
00973P 0270 39           EXIT4 RTS
```

```
00975                    *  THIS ROUTINE IS ENTERED IF THE ATTRIBUTE TABLE SHOWS A
00976                    *  FAULTY DEVICE AND THE ERROR HAS BEEN CONFIRMED.  A DUMMY
00977                    *  READ IS PERFORMED ON THE DATA REGISTER.  THIS IS THE
00978                    *  PROCEDURE FOR CLEARING A PENDING INTERRUPT.

00980P 0271 DE EF     8 EXIT3  LDX    IRQREC
00981P 0273 EE 06     A        LDX    6,X      FETCH THE DEVICE ADDRESS
00982P 0275 A6 01     A        LDAA   1,X      PERFORM A DUMMY READ ON DEVICE
00983P 0277 39                 RTS



00985                    *  THE ERROR ROUTINE (ACIAER) HAS DETECTED AN ERROR AND CONTROL
00986                    *  PASSES TO THIS ROUTINE.  IT IS ASSUMED THAT THE ERROR CAN NOT
00987                    *  BE CLEARED.  THEREFORE, THE DEVICE IS DECLARED FAULTY OR IN
00988                    *  ERROR.  THE ERROR COUNTER IS INCREMENTED AND THE CONTROL
00989                    *  REGISTER IS ALTERED.  THE RECEIVER INTERRUPT IS DISABLED.

00991P 0278 7C 03C0   D ACIAER INC    IRQBIT   INCREMENT ERROR COUNTER
00992P 0278 4F                 CLRA             SET MODIFICATION WORD
00993P 027C 6C 01     A        INC    1,X      SET DEVICE ERROR ATTRIBUTE
00994P 027E 7E 025D   P        JMP    ACIASR   ALTER THE CONTROL WORD OF THE ACIA



00996                    *  THIS IS THE ERROR CHECKING ROUTINE.  BEFORE A WORD IS READ
00997                    *  FROM THE ACIA A CHECK IS MADE FOR A FRAME ERROR, OVERRUN ERROR
00998                    *  AND DATA-CARRIER-DETECT (DCD) ERROR.  THE CARRY BIT IS SET
00999                    *  IF A POSITIVE DETECTION IS MADE.

01001P 0281 0C           ACHECK CLC              CLEAR CARRY
01002P 0282 85 04     A        BITA   #$04      TEST FOR A DATA-CARRIER-DETECT (DCD) ERROR
01003P 0284 26 09 028F         BNE    ACERR     A D-C-D EXISTS IF THE VALUE IS NONZERO
01004P 0286 85 20     A        BITA   #$20      TEST FOR AN OVERRUN ERROR
01005P 0288 26 05 028F         BNE    ACERR     AN ERROR EXISTS IF THE VALUE IS NONZERO
01006P 028A 85 10     A        BITA   #$10      TEST FOR A FRAME ERROR
01007P 028C 26 01 028F         BNE    ACERR     AN ERROR EXISTS IS THE VALUE IS NONZERO
01008P 028E 39                 RTS

01010P 028F 0D           ACERR  SEC             SET CARRY IN THE PRESENCE OF AN ERROR
01011P 0290 39                 RTS
```

```
01013                   *  INTERRUPT CONTROL

01015                   *  THESE ROUTINES ARE ACCESSEO WHEN THE INTERRUPT STATE OF THE
01016                   *  ACIA HAS TO BE ALTEREO.  THE ENABLE AND OISABLE ROUTINES
01017                   *  FOR THE INPUT INTERRUPT ARE HANOLEO THROUGH THE SAME
0101B                   *  ROUTINE WHILE THE OUTPUT INTERRUPT HAS ITS OWN ROUTINE.

01020          0291  P ACENIN EQU    *
01021P 0291 4F                 CLRA            SET ENABLE WORO
01022P 0292 BO 06 029A         BSR    ACIAIN   CALL INPUT INTERRUPT MOOIFER
01023P 0294 39                 RTS


01025                   *  THE OISABLE ROUTINE IS HANOLEO IN THE SAME MANNER

01027          0295  P ACOSIN EQU    *
0102BP 0295 B6 01    A         LOAA   #1       SET OISABLE INTERRUPT WORO
01029P 0297 BO 01 029A         BSR    ACIAIN   CALL INPUT INTERRUPT MOOIFIER
0103OP 0299 39                 RTS


01032                   *  THIS IS THE COMMON SUBROUTINE FOR THE INPUT SIOE OF THE ACIA.
01033                   *  THE RECEIVING INTERRUPT IS OISABLEO BY INHIBITTING THE
01034                   *  REQUEST-TO-SENO (RTS) LINE OF THE ACIA.  THUS, OISALLOWING
01035                   *  FURTHER TRANSMISSIONS BY THE OTHER ACIA'S.  THE ATTRIBUTE
01036                   *  TABLE IS ALSO MOOIFIEO TO REFLECT THIS ACTION.

0103BP 029A OE EF    B ACIAIN LOX    IRQREC
01039P 029C EE OA    A         LOX    1O,X     FETCH THE RTS PORT AOORESS
0104OP 029E A7 OO    A         STAA   O,X      UPOATE THE RTS PORT

01042P 02AO OE EF    B         LOX    IRQREC
01043P 02A2 EE OB    A         LOX    B,X      FETCH THE ATTRIBUTE TABLE POINTER
01044P 02A4 E6 OO    A         LOAB   O,X      FETCH WORO ZERO OF THE ATTRIBUTES
01045P 02A6 C4 FE    A         ANOB   #$FE
01046P 02AB 1B                 ABA             MOOIFY CONTROL WORO TO REFLECT RTS
01047P 02A9 A7 OO    A         STAA   O,X      UPOATE THE ATTRIBUTE TABLE
0104BP 02AB 39                 RTS
```

```
01050                    *  THESE ROUTINES ARE USED TO MODIFY THE OUTPUT INTERRUPT OR
01051                    *  THE TRANSMITTING SIDE OF AN ACIA.  ENABLING AND DISABLING THE
01052                    *  INTERRUPTS ARE ACHIEVED THROUGH THE SAME ROUTINE.

01054           02AC  P ACENOT EQU    *
01055P 02AC BD 0C 02BA         BSR    PREACI   FETCH THE CONTROL WORD AND ZERO OUT XMITTER
01056P 02AE 8A 20    A         ORAA   #$20     ENABLE TRANSMITTER INTERRUPT
01057P 02B0 BD 0D 02BF         BSR    POSTAC   UPDATE CONTROL AND ATTRIBUTE REGISTERS
01058P 02B2 39                 RTS


01060                    *  THIS ROUTINE WILL DISABLE THE TRANSMITTER
01061                    *  INTERRUPT AND UPDATE THE NECESSARY REGISTERS.

01063           02B3  P ACDSOT EQU    *
01064P 02B3 BD 05 02BA         BSR    PREACI   FETCH THE CONTROL WORD AND ZERO OUT TRANSMITTER
01065P 02B5 8A 40    A         ORAA   #$40     DISABLE TRANSMITTER INTERRUPT
01066P 02B7 BD 06 02BF         BSR    POSTAC   UPDATE THE CONTROL ANO ATTRIBUTE REGISTERS
01067P 02B9 39                 RTS


01069                    *  THIS ROUTINE FETCHES THE CONTROL WORD FOR THE ACIA
01070                    *  AND STRIPS OFF THE TRANSMITTER BITS.

01072P 02BA B6 D5    A PREACI LDAA   #ACNTRL  FETCH CONTROL WORD
01073P 02BC B4 9F    A         ANDA   #$9F     STRIP OFF TRANSMITTER BITS
01074P 02BE 39                 RTS


01076                    *  THIS IS THE MAIN ROUTINE FOR MODIFING THE OUTPUT OR TRANSMITTER
01077                    *  SIDE OF THE ACIA.  THIS ROUTINE WILL UPDATE THE ACIA AND
01078                    *  MODIFY THE ATTRIBUTE TABLE ACCORDINGLY.

01080P 02BF DE EF    B POSTAC LDX    IRQREC
01081P 02C1 EE 06    A         LDX    6,X      FETCH DEVICE ADDRESS
01082P 02C3 A7 00    A         STAA   0,X      ISSUE UPDATED CONTROL WORD TO ACIA
01083P 02C5 DE EF    B         LDX    IRQREC
01084P 02C7 EE 08    A         LDX    8,X      FETCH ATTRIBUTE TABLE POINTER
01085P 02C9 E6 00    A         LDAB   0,X      FETCH ATTRIBUTE WORD ZERO
01086P 02CB C4 7F    A         ANDB   #$7F     DISABLE TRANSMITTER INTERRUPT
01087P 02CD B5 40    A         BITA   #$40     TEST FOR ENABLE OR DISABLE
01088P 02CF 26 02 02D3         BNE    *+4      VALUE IS ZERO IF INTERRUPT IS DISABLED
01089P 02D1 CA 80    A         ORAB   #$80     ENABLE TRANSMITTING INTERRUPT
01090P 02D3 E7 00    A         STAB   0,X      UPDATE ATTRIBUTE TABLE
01091P 02D5 39                 RTS
```

```
01093                ************************************************************************
01094                *                                                                      *
01095                *    QUEUING ROUTINES                                                  *
01096                *                                                                      *
01097                *    THE FOUR ROUTINES (XFERIN), (XFEROT), (GETCHR), AND (PUTCHR) ARE *
01098                *    DESIGNED TO PERFORM DEVICE I/O UNDER A MULTI-TASKING ENVIONMENT. *
01099                *    (XFERIN) AND (XFEROT) ARE EXECUTED THROUGH THE DEVICE SERVICE    *
01100                *    ROUTINE WHICH, IN TURN, IS CALLED FROM THE INTERRUPT HANDLER.    *
01101                *    THE OTHER TWO ROUTINES ARE USED BY THE PROGRAMMER TO ISSUE       *
01102                *    AND ACCEPT DATA FROM DEVICE PORTS.  THERE ARE THREE MACROS       *
01103                *    WHICH COMPLETELY DEFINE THE QUEUING AND DEQUEUING OPERATIONS.    *
01104                *                                                                      *
01105                ************************************************************************



01107                *  MACRO: PREQU1

01109                *  THIS MACRO WILL DERIVE AND INITIALIZE THE POINTERS (IRQVAL) AND
01110                *  (QUEREC) FROM THE POINTER (IRQREC).  THIS ACTION CREATES
01111                *  SYMMETRY BETWEEN TWO ROUTINES.  THIS MACRO IS USED
01112                *  BY THE ROUTINES (XFERIN) AND (XFEROT).

01114                PREQU1 MACR   C
01115                  LDX IRQREC  FETCH THE DEVICE DEFINITION POINTER
01116                  IFNE FLAG
01117                  INX  INCREMENT VALUE BY ONE
01118                  ENDC
01119                  STX IRQVAL  INITIALIZE FROM (IRQREC)
01120                  SPC 1
01121                  LDA A 3,X  FETCH (QUEVAL) ADDRESS
01122                  IFNE FLAG
01123                  ADD A #$07  INCREASE POINTER BY SEVEN BYTES
01124                  ENDC
01125                  STA A QUEREC+1  INITIALIZE FROM (IRQREC)
01126                  ENDM
```

```
0112B                   *  MACRO: PREQU2

01130                   *  THIS MACRO WILL COMPUTE THE VALUES (IRQREC), (IRQVAL), AND
01131                   *  (QUEREC) FROM A DEVICE NUMBER.  THIS MACRO IS USED BY
01132                   *  THE (GETCHR) AND (PUTCHR) ROUTINES.

01134                   PREQU2 MACR   C
01135                     TAB
01136                     ASL A
01137                     ABA
0113B                     ASL A
01139                     ASL A  MULTIPLY DEVICE NUMBER BY THREE
01140                     ADD A IRQLST+1
01141                     STA A IRQREC+1  ADD DISPLACEMENT TO BASE ADDRESS
01142                     IFNE FLAG
01143                     INC A
01144                     ENDC
01145                     STA A IRQVAL+1  INITIALIZE FROM (IRQREC)
01146                     SPC 1
01147                     LDX IRQREC
0114B                     LDA A 3,X  FETCH (VARQUE) ADDRESS
01149                     IFNE FLAG
01150                     ADD A #$07  ADJUST TO POINT TO OUTPUT QUEUE
01151                     ENDC
01152                     STA A QUEREC+1  INITIALIZED
01153                     ENDM
```

```
01155                 *  MACRO:  QUDQUE

01157                 *  THIS IS THE MAIN MACRO WHICH IS USED 8Y ALL FOUR ROUTINES.
01158                 *  THE MACRO WILL FIRST TEST THE QUEUE COUNT.  IT WILL EITHER
01159                 *  CHECK FOR A MAXIMUM OR MINIMUM VALUE.  THE MACRO WILL
01160                 *  EITHER 8LOCK A TASK OR READY A 8LOCKED TASK.  THE COUNT
01161                 *  IS TESTED AGAIN AND THE INTERRUPT CONTROL ROUTINE IS ENTERED.
01162                 *  THE INTERRUPT ROUTINE IS ONE OF FOUR ROUTINES FOUND IN THE
01163                 *  DEVICE SERVICE ROUTINE SECTION.  THE FINAL TASK OF THIS
01164                 *  MACRO IS TO EITHER QUEUE OR DEQUE DATA ACCORDINLY.

01166                 QUDQUE MACR   C
01167                   LDX QUEREC
01168                   LDA A 6,X  FETCH QUEUE COUNTER ADDRESS
01169                   CMP A #1  TEST QUEUE COUNTER
01170                   8NE *+5  TEST FAILS IF VALUE IS NONZERO
01171                   JSR 2  ROUTINE IS EXECUTED IF VALUE IS ZERO
01172                   SPC 1
01173                   LDX QUEREC
01174                   LDA A 6,X  FETCH QUEUE COUNTER ADDRESS
01175                   CMP A #3  TEST QUEUE COUNTER
01176                   8NE *+12  TEST FAILS IF VALUE IS NONZERO
01177                   SPC 1
01178                   PSH A
01179                   LDX IRQREC
01180                   LDX 4,X  FETCH DEVICE SERVICE ROUTINE ADDRESS
01181                   LDX 4,X  FETCH APPROPRIATE INTERRUPT CONTROL VECTOR
01182                   JSR 0,X  8RANCH TO INTERRUPT CONTROL ROUTINE
01183                   PUL A
01184                   SPC 1
01185                   LDX QUEREC
01186                   0 A
01187                   STA A 6,X  UPDATE QUEUE COUNTER
01188                   JSR 5  EXECUTE PRIMARY FUNCTION
01189                   ENDM
```

```
01191                    *  TRANSPARENT QUEUING

01193                    *  WHEN A DEVICE REQUIRES SERVICE AND THERE IS INCOMING DATA
01194                    *  THIS ROUTINE IS ENTERED.  THE OPTIONS OF THE MACRO (PREQU1) ARE
01195                    *  NOT TAKEN.  THE ARGUMENTS STATE THAT IF THE QUEUE COUNTER
01196                    *  REACHES ZERO, THE CORRESPONDING BLOCKED TASK IS READIED.
01197                    *  IN ADDITION, IF THE COUNT SHOULD REACH 13, THE INTERRUPT
0119B                    *  CONTROL ROUTINE IS ENTERED.  THIS ROUTINE IS CALL FROM
01199                    *  THE DEVICE SERVICE HANDLER.

01201           02D6  P XFERIN EQU    *
01202           0000  A FLAG   SET    0         REJECT THE OPTIONS OF (PREQU1)
01203P 02D6                    PREQU1 CALL     MACRO FOR POINTER SETUP
     P 02D6 DE EF     B        LDX    IRQREC   FETCH THE DEVICE DEFINITION POINTER
                0000  A        IFNE   FLAG
01204                       ENDC
     P 02DB DF F1     B        STX    IRQVAL   INITIALIZE FROM (IRQREC)

     P 02DA A6 03     A        LDAA   3,X      FETCH (QUEVAL) ADDRESS
                0000  A        IFNE   FLAG
01205                       ENDC
     P 02DC 97 F4     B        STAA   QUEREC+1 INITIALIZE FROM (IRQREC)
01206P 02DE                    QUDQUE INC,0,QREADY,13,2,QUEUE CALL MACRO TO DO THE WORK
     P 02DE DE F3     B        LDX    QUEREC
     P 02E0 A6 06     A        LDAA   6,X      FETCH QUEUE COUNTER ADDRESS
     P 02E2 B1 00     A        CMPA   #0       TEST QUEUE COUNTER
     P 02E4 26 03 02E9         BNE    *+5      TEST FAILS IF VALUE IS NONZERO
     P 02E6 BD 03C2   P        JSR    QREADY   ROUTINE IS EXECUTED IF VALUE IS ZERO

     P 02E9 DE F3     B        LDX    QUEREC
     P 02EB A6 06     A        LDAA   6,X      FETCH QUEUE COUNTER ADDRESS
     P 02ED B1 0D     A        CMPA   #13      TEST QUEUE COUNTER
     P 02EF 26 0A 02FB         BNE    *+12     TEST FAILS IF VALUE IS NONZERO

     P 02F1 36                 PSHA
     P 02F2 DE EF     B        LDX    IRQREC
     P 02F4 EE 04     A        LDX    4,X      FETCH DEVICE SERVICE ROUTINE ADDRESS
     P 02F6 EE 02     A        LDX    2,X      FETCH APPROPRIATE INTERRUPT CONTROL VECTOR
     P 02FB AD 00     A        JSR    0,X      BRANCH TO INTERRUPT CONTROL ROUTINE
     P 02FA 32                 PULA

     P 02FB DE F3     B        LDX    QUEREC
     P 02FD 4C                 INCA
     P 02FE A7 06     A        STAA   6,X      UPDATE QUEUE COUNTER
     P 0300 BD 03F5   P        JSR    QUEUE    EXECUTE PRIMARY FUNCTION
01207P 0303 39                 RTS
```

```
01209                    *  TRANSPARENT DEQUEUING

01211                    *  WHEN A DEVICE REQUIRES SERVICE AND DATA HAS TO BE ISSUED, THIS
01212                    *  ROUTINE IS CALLED.  THE OPTIONS OF THE MACRO (PREQU1) ARE
01213                    *  TAKEN.  THE ARGUMENTS STATE THAT IF THE COUNTER REACHES 13 THEN
01214                    *  THE BOCKED TASK WILL BE READIED.  ALSO, IF THE COUNTER REACHES
01215                    *  1 THE INTERRUPT CONTROL ROUTINE IS ENTERED.

01217              0304  P XFEROT EQU    *
0121B              0001  A FLAG   SET    1         ENABLE THE OPTIONS OF (PREQU1)
01219P 0304                      PREQU1 CALL     MACRO FOR POINTER SETUP
     P 0304 DE EF     B         LDX    IRQREC   FETCH THE DEVICE DEFINITION POINTER
                0001  A         IFNE   FLAG
     P 0306 08                  INX             INCREMENT VALUE BY ONE
                                ENDC
     P 0307 DF F1     B         STX    IRQVAL   INITIALIZE FROM (IRQREC)

     P 0309 A6 03     A         LDAA   3,X      FETCH (QUEVAL) ADDRESS
                0001  A         IFNE   FLAG
     P 030B BB 07     A         ADDA   #$07     INCREASE POINTER BY SEVEN BYTES
                                ENDC
     P 030D 97 F4     B         STAA   QUEREC+1 INITIALIZE FROM (IRQREC)
01220P 030F                      QUDQUE DEC,13,QREADY,1,6,DEQUE CALL MACRO TO DO THE WORK
     P 030F DE F3     B         LDX    QUEREC
     P 0311 A6 06     A         LDAA   6,X      FETCH QUEUE COUNTER ADDRESS
     P 0313 B1 0D     A         CMPA   #13      TEST QUEUE COUNTER
     P 0315 26 03 031A          BNE    *+5      TEST FAILS IF VALUE IS NONZERO
     P 0317 BD 03C2   P         JSR    QREADY   ROUTINE IS EXECUTED IF VALUE IS ZERO

     P 031A DE F3     B         LDX    QUEREC
     P 031C A6 06     A         LDAA   6,X      FETCH QUEUE COUNTER ADDRESS
     P 031E B1 01     A         CMPA   #1       TEST QUEUE COUNTER
     P 0320 26 0A 032C          BNE    *+12     TEST FAILS IF VALUE IS NONZERO

     P 0322 36                  PSHA
     P 0323 DE EF     B         LDX    IRQREC
     P 0325 EE 04     A         LDX    4,X      FETCH DEVICE SERVICE ROUTINE ADDRESS
     P 0327 EE 06     A         LDX    6,X      FETCH APPROPRIATE INTERRUPT CONTROL VECTOR
     P 0329 AD 00     A         JSR    0,X      BRANCH TO INTERRUPT CONTROL ROUTINE
     P 032B 32                  PULA

     P 032C DE F3     B         LDX    QUEREC
     P 032E 4A                  DECA
     P 032F A7 06     A         STAA   6,X      UPDATE QUEUE COUNTER
     P 0331 BD 040F   P         JSR    DEQUE    EXECUTE PRIMARY FUNCTION
01221P 0334 39                   RTS
```

```
01223                    *  TASK OEQUEUING

01225                    *  THIS ROUTINE IS CALLEO FROM A TASK WHEN A OATA BYTE IS TO BE
01226                    *  RECEIVEO.  ACCUMULATOR B CONTAINS THE OEVICE NUMBER ANO IS
01227                    *  UNEFFECTEO UPON EXIT.  ACCUMULATOR A WILL RECEIVE THE OATA BYTE.
01228                    *  WHEN THIS ROUTINE IS ENTEREO ANO THERE IS NO OATA BYTE AVAILABLE,
01229                    *  THE TASK MAKING THE CALL IS BLOCKEO UNTIL A OATA BYTE IS RECEIVEO.
01230                    *  THE ARGUMENTS OF THE MACRO (QUOQUE) STATE THAT WHEN THE QUEUE
01231                    *  COUNT BECOMES ZERO, THE CALLING TASK IS BLOCKEO.  THE INTERRUPT
01232                    *  CONTROL ROUTINE IS ENTEREO WHEN THE COUNT REACHES 13.

01234           0335  P GETCHR EQU    *
01235P 0335 36                 PSHA
01236P 0336 OF                 SEI
01237P 0337 7C 00F6  B         INC    CRTFLG   ENTERING CRITICAL REGION
0123B           0000  A FLAG   SET    0        IGNORE OPTIONS IN (PREQU2)
01239P 033A                    PREQU2 CALL     MACRO TO SET UP POINTERS
     P 033A 16                 TAB
     P 033B 4B                 ASLA
     P 033C 1B                 ABA
     P 033O 48                 ASLA
     P 033E 48                 ASLA           MULTIPLY OEVICE NUMBER BY THREE
     P 033F BB 0031  P         AOOA   IRQLST+1
     P 0342 97 FO    B         STAA   IRQREC+1 AOO OISPLACEMENT TO BASE AOORESS
               0000  A         IFNE   FLAG
01240                       ENOC
     P 0344 97 F2    B         STAA   IRQVAL+1 INITIALIZE FROM (IRQREC)

     P 0346 OE EF    B         LOX    IRQREC
     P 0348 A6 03    A         LOAA   3,X      FETCH (VARQUE) AOORESS
               0000  A         IFNE   FLAG
01241                       ENOC
     P 034A 97 F4    B         STAA   QUEREC+1 INITIALIZEO
```

```
01243P 034C                    QUOQUE OEC,0,QBLOCK,13,0,DEQUE CALL MACRO TO DO THE WORK
     P 034C DE F3    B         LDX    QUEREC
     P 034E A6 06    A         LDAA   6,X       FETCH QUEUE COUNTER ADDRESS
     P 0350 81 00    A         CMPA   #0        TEST QUEUE COUNTER
     P 0352 26 03 0357         BNE    *+5       TEST FAILS IF VALUE IS NONZERO
     P 0354 BD 03CF  P         JSR    QBLOCK    ROUTINE IS EXECUTED IF VALUE IS ZERO

     P 0357 DE F3    B         LDX    QUEREC
     P 0359 A6 06    A         LDAA   6,X       FETCH QUEUE COUNTER ADDRESS
     P 035B 81 0D    A         CMPA   #13       TEST QUEUE COUNTER
     P 035D 26 0A 0369         BNE    *+12      TEST FAILS IF VALUE IS NONZERO

     P 035F 36                 PSHA
     P 0360 DE EF    B         LDX    IRQREC
     P 0362 EE 04    A         LDX    4,X       FETCH DEVICE SERVICE ROUTINE ADDRESS
     P 0364 EE 00    A         LDX    0,X       FETCH APPROPRIATE INTERRUPT CONTROL VECTOR
     P 0366 AD 00    A         JSR    0,X       BRANCH TO INTERRUPT CONTROL ROUTINE
     P 036B 32                 PULA

     P 0369 DE F3    B         LDX    QUEREC
     P 036B 4A                 DECA
     P 036C A7 06    A         STAA   6,X       UPOATE QUEUE COUNTER
     P 036E BD 040F  P         JSR    DEQUE     EXECUTE PRIMARY FUNCTION
01244P 0371 D6 F5    B         LDAB   DATA
01245P 0373 7F 00F6  B         CLR    CRTFLG    LEAVING CRITICAL REGION
01246P 0376 0E                 CLI
01247P 0377 32                 PULA
0124BP 0378 39                 RTS
```

```
01250                    *  TASK QUEUING

01252                    *  THIS ROUTINE IS CALLED FROM A TASK WHEN A DATA BYTE IS TO BE
01253                    *  TRANSMITTED TO A DEVICE.  ACCUMULATOR B CONTAINS THE DEVICE
01254                    *  NUMBER AND IS UNEFFECTED UPON EXIT.  ACCUMULATOR A CONTAINS
01255                    *  THE DATA BYTE TO BE TRANSMITTED AND IS ALSO UNEFFECTED UPON
01256                    *  EXIT.  THE ARGUMENTS FOR THE MACRO (QUOQUE) STATE THAT WHEN
01257                    *  THE QUEUE COUNT IS 16, THE CALLING TASK WILL BE BLOCKED.  WHEN
0125B                    *  THE QUEUE COUNT IS 0, THE INTERRUPT CONTROL ROUTINE IS ENTERED.

01260          0379  P PUTCHR EQU    *
01261P 0379 36                 PSHA
01262P 037A 37                 PSHB            RETAIN REGISTERS
01263P 037B 0F                 SEI
01264P 037C 7C 00F6  B         INC     CRTFLG  ENTERING CRITICAL REGION
01265P 037F 07 F5    B         STAB    DATA
01266          0001  A FLAG    SET     1       ACCEPT THE OPTIONS OF (PREQU2)
01267P 0381                    PREQU2  CALL    MACRO TO SET UP POINTERS
     P 0381 16                 TAB
     P 0382 48                 ASLA
     P 0383 1B                 ABA
     P 0384 48                 ASLA
     P 0385 48                 ASLA            MULTIPLY DEVICE NUMBER BY THREE
     P 0386 BB 0031  P         ADDA    IRQLST+1
     P 0389 97 F0    B         STAA    IRQREC+1 ADD DISPLACEMENT TO BASE ADDRESS
               0001  A         IFNE    FLAG
     P 038B 4C                 INCA
                               ENDC
     P 038C 97 F2    B         STAA    IRQVAL+1 INITIALIZE FROM (IRQREC)

     P 038E 0E EF    B         LDX     IRQREC
     P 0390 A6 03    A         LDAA    3,X     FETCH (VARQUE) ADDRESS
               0001  A         IFNE    FLAG
     P 0392 BB 07    A         ADDA    #$07    ADJUST TO POINT TO OUTPUT QUEUE
                               ENDC
     P 0394 97 F4    B         STAA    QUEREC+1 INITIALIZED
```

```
01269P 0396                    QUOQUE INC,16,QBLOCK,0,4,QUEUE CALL MACRO TO 00 THE WORK
     P 0396 0E F3    B         LOX    QUEREC
     P 0398 A6 06    A         LOAA   6,X       FETCH QUEUE COUNTER AOORESS
     P 039A 81 10    A         CMPA   #16       TEST QUEUE COUNTER
     P 039C 26 03 03A1         BNE    *+5       TEST FAILS IF VALUE IS NONZERO
     P 039E B0 03CF  P         JSR    QBLOCK    ROUTINE IS EXECUTED IF VALUE IS ZERO

     P 03A1 0E F3    B         LOX    QUEREC
     P 03A3 A6 06    A         LOAA   6,X       FETCH QUEUE COUNTER AOORESS
     P 03A5 81 00    A         CMPA   #0        TEST QUEUE COUNTER
     P 03A7 26 0A 03B3         BNE    *+12      TEST FAILS IF VALUE IS NONZERO

     P 03A9 36                 PSHA
     P 03AA 0E EF    B         LOX    IRQREC
     P 03AC EE 04    A         LOX    4,X       FETCH OEVICE SERVICE ROUTINE AOORESS
     P 03AE EE 04    A         LOX    4,X       FETCH APPROPRIATE INTERRUPT CONTROL VECTOR
     P 03B0 A0 00    A         JSR    0,X       BRANCH TO INTERRUPT CONTROL ROUTINE
     P 03B2 32                 PULA

     P 03B3 0E F3    B         LOX    QUEREC
     P 03B5 4C                 INCA
     P 03B6 A7 06    A         STAA   6,X       UPOATE QUEUE COUNTER
     P 03B8 B0 03F5  P         JSR    QUEUE     EXECUTE PRIMARY FUNCTION
01270P 03BB 7F 00F6  B         CLR    CRTFLG    LEAVING CRITICAL REGION
01271P 03BE 0E                 CLI
01272P 03BF 33                 PULB
01273P 03C0 32                 PULA              RESTORE REGISTERS
01274P 03C1 39                 RTS
```

```
01276                    *  READY A TASK

01278                    *  THIS ROUTINE WILL ENABLE AN ALREADY EXISTING BLOCKED TASK.
01279                    *  A TASK IS KNOWN BLOCKED IF THE BYTE POSITIONS 4 AND 5 (VARQUE)
01280                    *  CONTAIN THE TASK STACK POINTER.  OTHERWISE, THE VALUE IS
01281                    *  ZERO AND, THUS, THE TASK IS READY.

01283          03C2  P QREADY EQU    *
01284P 03C2 DE F3    B        LDX    QUEREC
01285P 03C4 EE 04    A        LDX    4,X      FETCH THE BLOCKED TASK ADDRESS
01286P 03C6 27 06 03CE        BEQ    NOBLCK   VALUE IS ZERO IF NO TASK IS BLOCKED
01287P 03C8 6F 02    A        CLR    2,X      READY TASK

01289P 03CA DE F3    B        LDX    QUEREC
01290P 03CC 6F 05    A        CLR    5,X      CLEAR BLOCK TASK ADDRESS
01291P 03CE 39         NOBLCK RTS
```

```
01293                    *  BLOCK A TASK

01295                    *  THIS ROUTINE WILL BLOCK THE TASK CALLING IT.  THE VARIABLES
01296                    *  (IRQREC), (IRQVAL), (QUEREC), (DATA), (CRTFLG) ARE STACKED
01297                    *  FOR FUTURE RETRIEVAL.  THE CURRENT TASK IS BLOCKED AND A SWI
01298                    *  INTERRUPT IS GENERATED.  CONTROL WILL PASS TO THE SCHEDULER.
01299                    *  WHEN THE TASK IS UNBLOCKED, THE ABOVE MENTIONED VARIABLES
01300                    *  WILL BE RETRIEVED AND REINSTATED.  POINTERS IN THE BASE
01301                    *  PAGE ARE ONE BYTE IN LENGTH.

01303          03CF   P QBLOCK EQU    *
01304P 03CF 36                 PSHA
01305P 0300 DE F3      B       LDX    QUEREC
01306P 0302 D6 EE      B       LDAB   TSKPTR+1 FETCH TASK POINTER ADDRESS
01307P 0304 E7 05      A       STAB   5,X      BLOCK TASK BY INSERTING STACK POINTER IN (VARQUE)
01308P 0306 DE E0      B       LDX    TSKPTR   FETCH TASK POINTER ADDRESS
01309P 0308 C6 B0      A       LDAB   #$B0

01311P 030A C6 0B      A       LDAB   #8
01312P 030C CE 00EF    B       LDX    #IRQREC
01313P 030F A6 00      A LOOP1 LDAA   0,X
01314P 03E1 36                 PSHA
01315P 03E2 08                 INX
01316P 03E3 5A                 DECB
01317P 03E4 26 F9 030F         BNE    LOOP1    REPEAT EIGHT TIMES

01319P 03E6 3F                 SWI             GENERATE SOFTWARE INTERRUPT


01321                    *  RETRIEVE AND RESTORE REGISTERS

01323P 03E7 C6 08      A       LDAB   #8
01324P 03E9 CE 00F6    B       LDX    #IRQREC+7
01325P 03EC 32           LOOP2 PULA
01326P 03ED A7 00      A       STAA   0,X
01327P 03EF 09                 DEX
01328P 03F0 5A                 DECB
01329P 03F1 26 F9 03EC         BNE    LOOP2
01330P 03F3 32                 PULA            RESTORE ACCUMULATOR
01331P 03F4 39                 RTS
```

```
01333                 *  MACRO: FIXPTR

01335                 *  THE QUEUING AND OEQUEUING ROUTINES ARE PRACTICALLY IDENTICAL.
01336                 *  THIS MACRO WILL PERFORM THE INCREMENTING OF THE POINTERS ON A
01337                 *  MODULO 16 BASES.  THE POINTERS ARE UPDATED ACCORDINGLY.  THE
01338                 *  VARIATION BETWEEN THE TWO ROUTINES IS THE POINTER IN QUESTION.
01339                 *  THE DIFFERENCE IS PASSED AS AN ARGUMENT REPRESENTING AN
01340                 *  INDEXED DISPLACEMENT VALUE.

01342                 FIXPTR MACR   C
01343                   LDX QUEREC
01344                   LDA A 0,X  FETCH POINTER
01345                   INC A  INCREMENT POINTER
01346                   LDX IRQVAL
01347                   CMP A 1,X  CHECK FOR MAXIMUM RESULT
0134B                   BNE *+4  TEST IS POSITIVE IF VALUE IS ZERO
01349                   LDA A 0,X  RESET POINTER VALUE
01350                   LDX QUEREC
01351                   STA A 0,X  REINSTATE POINTER
01352                   ENDM
```

```
01354                    *  QUEUING ROUTINE

01356                    *  THIS ROUTINE IS CALLED TO QUEUE A DATA BYTE.  DATA IS STORED
01357                    *  AT THE CURRENT [REAR] POINTER LOCATED IN THE (VARQUE) TABLE.  THE
0135B                    *  POINTER IS INCREASED TO THE NEXT VALUE (MODULO 16) AND THE COUNTER
01359                    *  INCREMENTED BY ONE.

01361           03F5  P QUEUE  EQU    *
01362P 03F5 06 F5     B        LDAB   DATA
01363P 03F7 0E F3     B        LDX    QUEREC
01364P 03F9 EE 02     A        LDX    2,X       FETCH REAR ADDRESS POINTER
01365P 03FB E7 00     A        STAB   0,X       QUEUE DATA

01367P 03FD                    FIXPTR 3         CALL MACRO TO INCREMENT POINTER
     P 03FD 0E F3     B        LDX    QUEREC
     P 03FF A6 03     A        LDAA   3,X       FETCH POINTER
     P 0401 4C                INCA             INCREMENT POINTER
     P 0402 0E F1     B        LDX    IRQVAL
     P 0404 A1 01     A        CMPA   1,X       CHECK FOR MAXIMUM RESULT
     P 0406 26 02 040A        BNE    *+4       TEST IS POSITIVE IF VALUE IS ZERO
     P 0408 A6 00     A        LDAA   0,X       RESET POINTER VALUE
     P 040A 0E F3     B        LDX    QUEREC
     P 040C A7 03     A        STAA   3,X       REINSTATE POINTER
0136BP 040E 39                 RTS
```

```
01370                    *  DEQUEUING ROUTINE

01372                    *  THIS ROUTINE IS CALLED TO DEQUEUE A DATA BYTE.  DATA IS
01373                    *  RETRIEVED FROM THE CURRENT [FRONT] POINTER LOCATED IN THE
01374                    *  (VARQUE) TABLE.  THE POINTER IS INCREASED TO POINT TO THE NEXT
01375                    *  BYTE (MODULO 16) AND THE COUNTER IS DECREMENTED BY ONE.

01377           040F  P DEQUE  EQU    *
0137BP 040F DE F3     B        LDX    QUEREC
01379P 0411 EE 00     A        LDX    0,X       FETCH FRONT POINTER AODRESS
013B0P 0413 E6 00     A        LDAB   0,X       DEQUEUE DATA
013B1P 0415 D7 F5     B        STAB   DATA

013B3P 0417                    FIXPTR 1         CALL MACRO TO INCREMENT POINTER
     P 0417 DE F3     B        LDX    QUEREC
     P 0419 A6 01     A        LDAA   1,X       FETCH POINTER
     P 041B 4C                 INCA             INCREMENT POINTER
     P 041C DE F1     B        LDX    IRQVAL
     P 041E A1 01     A        CMPA   1,X       CHECK FOR MAXIMUM RESULT
     P 0420 26 02 0424         BNE    *+4       TEST IS POSITIVE IF VALUE IS ZERO
     P 0422 A6 00     A        LDAA   0,X       RESET POINTER VALUE
     P 0424 DE F3     B        LDX    QUEREC
     P 0426 A7 01     A        STAA   1,X       REINSTATE POINTER
013B4P 042B 39                 RTS
```

```
01386                  ************************************************************************
01387                  *                                                                      *
01388                  *    SOFTWARE INTERRUPT HANDLER                                        *
01389                  *                                                                      *
01390                  *    AN SWI INSTRUCTION GENERATES AN INTERRUPT AS WOULD AN NMI OR      *
01391                  *    IRQ INTERRUPT.  THE REGISTERS ARE STACKED AND THE VECTOR AT       *
01392                  *    LOCATION $FFFA PASSES CONTROL TO THIS HANDLER.  THE PURPOSE       *
01393                  *    OF THIS ROUTINE IS TO RELINQUISH CONTROL FROM ONE TASK AND        *
01394                  *    PASS IT TO ANOTHER.  THE ROUTINE WILL SET THE (SKPFLG) AND        *
01395                  *    CLEAR THE (CRTFLG) FLAG.  AN ABSOLUTE JUMP IS PERFORMED INTO      *
01396                  *    THE TASK SCHEDULER ROUTINE.                                       *
01397                  *                                                                      *
01398                  ************************************************************************


01400           0429  P SWISVC EQU    *
01401P 0429 4F                 CLRA
01402P 042A 4C                 INCA
01403P 0428 97 FA     8        STAA   SKPFLG   SET FLAG
01404P 042D 4A                 DECA
01405P 042E 97 F6     8        STAA   CRTFLG   CLEAR CRITICAL REGION
01406P 0430 7E 0478   P        JMP    TSKCHG   PASS CONTROL TO SCHEDULER
```

```
0140B                   ************************************************************************
01409                   *                                                                      *
01410                   *    NON-MASKABLE INTERRUPT HANDLER                                    *
01411                   *                                                                      *
01412                   *    A REAL TIME CLOCK WITH A FREQUENCY OF 2152 HERTZ IS FED INTO THE *
01413                   *    NMI INPUT.  THIS ROUTINE INTERRUPTS THE GENERAL FLOW OF THE      *
01414                   *    PROGRAM AT THE RATED FREQUENCY AND THIS ROUTINE EXECUTES.        *
01415                   *    THERE ARE TWO MAJOR SECTIONS OF THIS ROUTINE.  THE FIRST DEALS   *
01416                   *    WITH COUNTING DOWN THE DELAYED TIMES OF EACH TASK, IF ANY, AND   *
01417                   *    THE SECOND HAS TO DO WITH TASK SCHEDULING AND DISPATCHING.  THE  *
0141B                   *    INITIAL FREQUENCY IS DIVIDE TWICE, ONCE BY THE SECONDARY TICKS   *
01419                   *    AND THE OTHER BY THE PRIMARY TICKS.  AFTER THE SECONDARY TICKS   *
01420                   *    HAVE EXPIRED, THE PRIORITY TASK IS INTERROGATED FOR ITS STATUS   *
01421                   *    AND ENTERED IF READIED.                                          *
01422                   *                                                                      *
01423                   ************************************************************************



01425                   *  CHECK THE (HOLDIT) FLAG TO DETERMINE WHETHER THE NMI ROUTINE
01426                   *  MAY BE ENTERED.  IF IT IS POSSIBLE, COUNT DOWN THE PRIMARY TICKS.
01427                   *  ONE TICK PER CLOCK PULSE.  EXIT IF COUNT IS NOT ZERO.

01429           0433  P NMISVC EQU    *
01430P 0433 96 FE     B        LDAA   HOLDIT   TEST FLAG
01431P 0435 26 05 043C         BNE    OFFITO   NMI ROUTINE IS ON IF VALUE IS ZERO

01433P 0437 7A 00F7   B        DEC    SUBVAL   COUNT DOWN SECONDARY TICKS
01434P 043A 27 01 043D         BEQ    NMIENT   CONTINUE IF COUNT REACHES ZERO
01435P 043C 3B          OFFITO RTI


01437                   *  REINSTATE THE SECONDARY VALUE.  TEST THE
0143B                   *  STATUS OF THE PRIORITY TASK AND ENTER IF READIED.

01440           043D  P NMIENT EQU    *
01441P 043D B6 00C7   P        LDAA   SUBTIC
01442P 0440 97 F7     B        STAA   SUBVAL   REINSTATE SECONDARY TICK VALUE

01444P 0442 CE 00E2   B        LDX    #PRITSK
01445P 0445 A6 02     A        LDAA   2,X      TEST STATUS WORD
01446P 0447 26 03 044C         BNE    DECPRI   TASK IS READY IF STATUS VALUE IS ZERO
01447P 0449 AE 00     A        LDS    0,X
0144BP 044B 3B                 RTI             ENTER PRIORITY TASK


01450                   *  SECONDARY TICKS HAVE EXPIRED.  COUNT DOWN THE PRIMARY TICKS.
01451                   *  ONE TICK PER CLOCK PULSE DIVIDED BY SECONDARY TICKS.

01453           044C  P DECPRI EQU    *
01454P 044C 7A 00FB   B        DEC    PRIVAL   COUND DOWN PRIMARY TICKS
01455P 044F 27 01 0452         BEQ    DECTIM   EXIT IF VALUE IS NOT ZERO
01456P 0451 3B                 RTI
```

```
01458                   *  DECREMENT DELAYS

01460                   *  DECREMENT THE DELAYS FOUND IN THE STATUS WORD OF EACH TASK.
01461                   *  IF THE WORD IS A POSITIVE VALUE, DECREMENT IT BY ONE.
01462                   *  OTHERWISE, PASS IT BY.  THIS TECHNIQUE IS USED TO DELAY
01463                   *  TASKS USING A GHOST OPERATOR TYPE SYSTEM.

01465           0452  P DECTIM EQU    *
01466P 0452 B6 00CB   P         LDAA   PRITIC
01467P 0455 97 FB     B         STAA   PRIVAL   REINSTATE THE PRIMARY TICK VALUE
0146BP 0457 CE 00BB   B         LDX    #PROCSS  FETCH TASK POINTER AND STATUS ADDRESS
01469P 045A C6 0F     A         LDAB   #NTASK+1 INITIALIZE FOR LOOP COUNT
0147DP 045C A6 02     A TIMELP  LDAA   2,X      TEST STATUS WORD FOR CONDITION
0147EP 045E 27 05 0465          BEQ    *+7      TASK READIED IF VALUE IS ZERO
0147FP 0460 2B 03 0465          BMI    *+5      TASK IS BLOCKED IF VALUE IS NEGITIVE
01473P 0462 4A                  DECA            DECREMT DELAY
01474P 0463 A7 02     A         STAA   2,X      UPDATE STATUS REGISTER
01475P 0465 08                  INX
01476P 0466 08                  INX
01477P 0467 08                  INX             POINT TO NEXT TASK POINTER
0147BP 0468 5A                  DECB            DECREMENT LOOP COUNTER
01479P 0469 26 F1 045C          BNE    TIMELP   REPEAT IF NECESSARY

01481                   *  IF THE PROCESSOR IS EXECUTING CODE WITHIN
01482                   *  A CRITICAL REGION,  DO NOT ENTER SCHEDULER.

01484P 046B 96 F6     B         LDAA   CRTFLG
01485P 046D 27 01 0470          BEQ    DIVB2    CRITICAL REGION IF VALUE IS NONZERO
01486P 046F 3B                  RTI


01488                   *  IF THE (SKPFLG) FLAG IS SET, CLEAR IT AND EXIT.
01489                   *  OTHERWISE, CONTINUE INTO THE SCHEDULER.

01491           0470  P DIVB2  EQU    *
01492P 0470 96 FA     B         LDAA   SKPFLG   TEST FLAG
01493P 0472 27 04 0478          BEQ    TSKCHG   FLAG IS CLEAR IF VALUE IS ZERO
01494P 0474 7F 00FA   B         CLR    SKPFLG   ALREADY SET FLAG, CLEAR IT
01495P 0477 3B                  RTI
```

```
01497                    *  TASK SCHEDULER

01499                    *  THIS IS THE SCHEDULER OF THE MONITOR.  CONTROL PASSES TO THIS
01500                    *  ROUTINE WHEN A NEW TASK IS TO TAKE CONTROL.  IF ALL TASKS ARE
01501                    *  BLOCKED, THE ROUTINE WILL LOOP UNTIL ONE IS READIED.  THE
01502                    *  EXITING TASK POINTER IS STORED IN THE TABLE (PROCSS).  THE
01503                    *  SAME TABLE IS SCANNED BY THE SCHEDULER TO FIND A SUCCESSOR.
01504                    *  FOR EACH SCAN, THE DEVICE ERROR BIT IS TESTED.  IF AN ERROR
01505                    *  EXISTS, THE (LOOKUP) ROUTINE WILL BE ENTERED TO LOCATED AND
01506                    *  CORRECT THE ERROR.  THE PROCESSOR INTERRUPT LINE IS ENABLED
01507                    *  BETWEEN TASK SEARCHES.  IF A PENDING INTERRUPT EXISTS,
01508                    *  IT WILL BE SERVICED AT THIS TIME.  THE FLAG (INTFLG) WILL
01509                    *  PREVENT TWO OR MORE CONTROL PATHS FROM ENTERING THIS AREA.

01511           047B  P TSKCHG EQU    *
01512P 047B 7C 00F9  B         INC    INTFLG   TEST FLAG FOR CRITICAL REGION
01513P 047B 26 36 04B3         BNE    EXIT2    SCHEDULER EMPTY IF FLAG IS ZERO

01515                    *  STORE STACK POINTER INTO TABLE.
01516                    *  THIS IS ACCOMPLISHED ONE BYTE AT A TIME.

0151BP 0470 BF 03C1  0         STS    TEMP1
01519P 04B0 0E E0    B         LDX    TSKPTR   FETCH SP DESTINATION ADDRESS
0152DP 04B2 F6 03C1  0         LDAB   TEMP1
01521P 04B5 E7 00    A         STAB   0,X      UPDATE MSB
01522P 04B7 F6 03C2  0         LDAB   TEMP1+1
01523P 04BA E7 01    A         STAB   1,X      UPDATE LSB
```

```
01525                    *  IF THE SCHEDULER HAS COMPLETED ONE SCAN OF THE TABLE, TEST
01526                    *  THE DEVICE ERROR BIT.  IF SET, ENTER THE (LOOKUP) TABLE.
01527                    *  ADVANCE THE SCHEDULER POINTER BY THREE.

01529P 04BC DE ED    B TSKAGN LDX    TSKPTR
01530P 04BE BC 00DF  B        CPX    #(NTASK-1)*3+PROCSS TEST FOR END OF TABLE
01531P 0491 26 0D 04A0        BNE    TSKCNU   END OF TABLE IF VALUE IS ZERO
01532P 0493 B6 03C0  D        LDAA   IRQBIT   TEST DEVICE ERROR BIT
01533P 0496 27 03 049B        BEQ    *+5      NO ERROR EXIST IF VALUE IS ZERO
01534P 049B BD 0190  P        JSR    LOOKUP   SEARCH FOR AND CORRECT THE ERROR
01535P 049B CE 00B7  B        LDX    #PROCSS-1 RESET THE TABLE POINTER
01536P 049E 20 02 04A2        BRA    *+4
01537P 04A0 08          TSKCNU INX
01538P 04A1 08                 INX
01539P 04A2 08                 INX             INCREMENT TABLE POINTER BY THREE
01540P 04A3 DF ED    B        STX    TSKPTR   UPDATE TABLE POINTER
01541P 04A5 A6 02    A        LDAA   2,X      TEST STATUS WORD OF TASK
01542P 04A7 27 0B 04B1        BEQ    CHANGE   TASK IS READIED IF VALUE IS ZERO
01543P 04A9 AE 00    A        LDS    0,X      ADVANCE TO NEXT STACK POSITION
01544P 04AB 0E                 CLI
01545P 04AC 01                 NOP             ALLOW PENDING INTERRUPTS TO OCCUR, BRIEFLY
01546P 04AD 01                 NOP
01547P 04AE 0F                 SEI
01548P 04AF 20 DB 048C        BRA    TSKAGN   LOOP WITHIN SCHEDULER

01550                    *  READIED TASK FOUND.  LOAD STACK POINTER
01551                    *  AND RETURN FROM INTERRUPT.  THIS PROCEDURE
01552                    *  WILL CAUSE THE TASK TO BE ENTERED.

01554P 04B1 AE 00    A CHANGE LDS    0,X
01555P 04B3 7A 00F9  B EXIT2  DEC    INTFLG   EXIT SCHEDULER
01556P 04B6 3B                 RTI
```

```
0155B                   *  STATUS CHANGE

01560                   *  THE (XCHSTS) ROUTINE MAY BE CALLED BY ANY TASK.  ITS
01561                   *  PURPOSE IS TO CHANGE THE STATUS OF ANY TASK.  SINCE THIS
01562                   *  ROUTINE MANIPULATES THE TASK SCHEDULING TABLES, IT OPERATES
01563                   *  UNDER THE CRITCIAL REGION MODE.  ACCUMULATOR B CONTAINS
01564                   *  THE NEW STATUS INFORMATION AND IS UNEFFECTED UPON EXIT.
01565                   *  ACCUMULATOR A CONTAINS THE TASK NUMBER AND IS UNDEFINED UPON
01566                   *  EXIT FROM THIS ROUTINE.

0156B          04B7  P XCHSTS EQU    *
0156BP 04B7 37                PSHB
0157OP 04B8 OF                SEI
0157lP 04B9 7C 00F6  B        INC    CRTFLG   SET CRITICAL REGION MODE
0157 2P 04BC 16               TAB
0157 3P 04BD 4B               ASLA
0157 4P 04BE 1B               ABA             MULTIPLY B BY THREE


01576                   *  THE CALCULATED VALUE BECOMES A DISPLACEMENT
01577                   *  AND IS ADDED TO THE INITIAL TABLE POINTER.
0157B                   *  THIS WILL GIVE THE ACTUAL TASK ADRRESS INTO THE TABLE.

015B0P 04BF CE 00BB  B        LDX    #PROCSS
015B1P 04C2 DF FB    B        STX    TEMP
015B2P 04C4 9B FC    B        ADDA   TEMP+1
015B3P 04C6 97 FC    B        STAA   TEMP+1
015B4P 04CB 24 03 04CD        BCC    *+5
015B5P 04CA 7C 00FB  B        INC    TEMP
015B6P 04CD DE FB    B        LDX    TEMP
015B7P 04CF 33                PULB
015B8P 04D0 E7 02    A        STAB   2,X      UPDATE STATUS REGISTER
015B9P 04D2 7A 00F6  B        DEC    CRTFLG   EXIT CRITICAL REGION
0159OP 04D5 0E                CLI
0159lP 04D6 39                RTS

01593                         END
TOTAL ERRORS 00000
```