A MICROPROCESSOR BASED STAR COMMUNICATION NETWORK

BY
DONALD GREGORY DRYGALSKI JR.
B.S., University of Illinois, 1977

THESIS
Submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering in the Graduate College of the University of Illinois at Urbana-Champaign 1979

Urbana, Illinois
I would like to thank the following people for their help and advice on this project:

Peter S. Moldauer
Mary K. Phinney
Andrew N. Fu
Ricardo B. Uribe
William G. Albright
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II</td>
<td>THE SYSTEM</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2.1 Connect Command</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2.2 Reconnect Command</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2.3 Status Command</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2.4 Disconnect Command</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2.5 Escape Command</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2.6 Message Command</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2.7 Baud Rate Command</td>
<td>7</td>
</tr>
<tr>
<td>III</td>
<td>HARDWARE</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>3.1 CPU Board</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>3.2 Memory Board</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>3.3 Serial I/O Board</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>3.4 Interrupt Board</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>3.5 Mother Board and the System Busses</td>
<td>27</td>
</tr>
<tr>
<td>IV</td>
<td>SOFTWARE</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>4.1 The System's Program</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>4.2 The System's Tables and Buffers</td>
<td>37</td>
</tr>
<tr>
<td>V</td>
<td>FUTURE MODIFICATIONS</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>5.1 Minor Software Modification</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>5.2 Minor Hardware Modification</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>5.3 Major System Modification</td>
<td>45</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

The Star Communication Network is a very useful piece of laboratory equipment because much time and effort can be saved when data links are automated. This network replaces matrices of rotary switches or of plug networks. A computer or terminal, that is connected to this network, is able to quickly connect to other devices since an operator does not have to physically change a connection. The star network is similar to one type of automated telephone switching system since both use a central processor to control the flow of data from any input to any other output. Thus, a star network increases the power of a small computer system since it can easily connect to a larger computer system that has, for example, a mass storage device. Also, the star network can increase the usage of expensive peripheral devices, because many systems can easily access these devices through the star network. The topic of this paper is the design of a star communication network.

This system must have the following requirements in order to be a useful tool. First, it must be easy to operate, because people will not want to learn a complicated computer language to do a simple task. If the commands are too difficult
to learn, people will avoid the use of this tool. Another requirement is that slower devices must be able to communicate with faster devices. Unlike the switch networks, the star network should let a 110 BAUD terminal communicate with a 9600 BAUD microprocessor. This means that the star network must buffer incoming data. Finally, the star network must be in control of all conversations. Two lines, the 'Data Set Ready' and the 'Data Terminal Ready' lines, must be used to control the devices on the system. If a device tries to send too much data to the network, the system must temporarily halt the device so that data is not lost.

This system is also designed such that it can easily be expanded. The initial star network is able to control 16 I/O ports, but this may not be enough for future use. Identical 8 port I/O boards can be added to the system to expand the size of the network. The number of buffers must also increase, since the number of ports can be increased. Identical 8K RAM boards can be added to this system to increase its storage capacity. Finally, speed will be a very important factor, since the system can grow larger. An appropriate method is needed to increase the speed of this system. The final chapter of this paper will briefly explain one way to do this.
CHAPTER II

USER COMMANDS

There are two operating modes for this system. In command mode, a user can execute any of the user commands. The user will receive a prompt sequence from the star network to show that he is in command mode. A carriage return indicates the end of the command. An error message is printed if an invalid command is entered. An invalid character is ignored by the star and will not be echoed back to the device if it is in command mode. All ports are placed in command mode when the system is powered up or is reset.

The second mode, connect mode, is used to transfer data from one port to another. The user will not be prompted by the star network in this mode. The user can go from connect mode to command mode by entering his command sequence. Initially, all ports have the 'ESC' key as their escape sequence. All but the last character of the escape sequence is passed on as data to the other port. A port can be prevented from disconnecting a link, by removing its escape sequence with one of the system commands. Most microprocessors should not be able to disconnect a link, so their escape sequence should be removed. Removal of this sequence will also increase the speed of the system. If a port receives an invalid character in connect mode, all the ports it is linked to will be put into
command mode and an 'Invalid Character' message will be printed on all ports.

There are two types of user commands for this system. The first type of command is for the common user. He should be able to do basic operations such as connecting his port to another port. These commands have a basic construction such as the connect command, 'C 0, 1'. These commands start with a one letter code and may have a list of decimal parameters following it. These parameters are always separated by a comma. The other type of command is for a responsible user that knows how to operate the system well. These commands can control ports other than the users port, so one must be careful when using them. These commands contain either a password or a special parameter, so they have a structure different from the general user commands. Also, if a parameter is invalid or out of range, an 'Invalid Command' message will be printed rather than an 'Invalid Parameter' message. This way the common user will not know that these commands exist if he is playing with the system.

2.1 Connect Command

C #1, #2, ...

The connect command is a general user command and is used to connect two or more ports. A minimum of two parameters is required for this command. The star network will take the output from the first port and input it to the second port. Likewise, the output of the second port is input to the third port and so on. The output from the final port is input to the
first port. All ports must be in command mode to be put into connect mode. If one of the ports is in connect mode, the 'Port Busy' message will be displayed, and all the unconnected ports will remain in command mode.

2.2 Reconnect Command   R

The reconnect command, another general user command, is useful when a connection is accidentally dropped or when the same connection is always requested. This command reconnects the users port in the same manner that it was previously connected. For example, if the user is on port 1, and he was connected to port 2, he could reconnect to port 2 by using the reconnect command. Again, port 2 must be in command mode for the connection to occur. If port 2 is not in command mode, port 1 will remain in command mode and the 'Port Busy' message will be printed on port 1.

2.3 Status Command   S #1, #2, . . . , #5

The status command, the final general user command, is used to print the status of part of the network. The user must input from one to five parameters for this command. The port number of the users terminal is the first piece of data to be displayed. A heading is displayed below this. Then one row of data is printed for each parameter that is requested. This data contains the escape sequence of that port in hexadecimal characters, the mode of operation, the input to that port and the receiver of that port's data.
2.4 Disconnect Command  D #, PLEASE

The disconnect command is used to disconnect a conversation that is external to the user's port. The requested port must be in connect mode, otherwise an error message will be printed. If that port is in connect mode, all ports in that conversation will be put into command mode and the prompt message will be sent to them. This command is followed by the system's password, 'PLEASE'. If the password is not present, an 'Invalid Command' message is output to the user's terminal.

2.5 Escape Command  X#, PLEASE, YYYYY

This command changes the escape code of the requested port. This port can be in either command mode or connect mode for the change to occur. The system password follows the port number, in order to prevent a general user from changing escape codes. The final parameter is optional. If there is no final parameter, the escape code is erased. This port will not be able to disconnect a command and will not be able to execute commands. The system will run somewhat faster if escape codes are erased. If the final parameter exists, the escape code is updated to the contents of this parameter. The maximum length of an escape sequence is six characters.
2.6 Message Command   M#, PLEASE, Message

The message command sends a message from the user's terminal to the requested port. This port can be in either connect mode or command mode. The port number must be followed by the system's password. The last parameter is the message that will be sent to the other terminal. A carriage return and line feed will automatically be placed at the end of this message.

2.7 Baud Rate Command   B#, XXXX

The baud rate command updates the baud rate of the requested port. The second parameter is the three or four character baud rate. The allowable baud rates are:

<table>
<thead>
<tr>
<th></th>
<th>110</th>
<th>300</th>
<th>600</th>
</tr>
</thead>
<tbody>
<tr>
<td>1200</td>
<td>2400</td>
<td>4800</td>
<td>9600</td>
</tr>
</tbody>
</table>

Two stop bits will be sent out for all 110 baud ports and only one stop bit will be sent out for all other rates.
CHAPTER III

SYSTEM HARDWARE

The hardware of the star network is broken up into four types of circuit boards that are connected by the system’s backplane. The CPU board controls the entire system. The addressing signals and most of the control signals originate from this board. The program memory in ROM is also located on this board. A second board, the 8K RAM board, holds the system's tables and buffers. There is enough storage space for a twenty-four port system on this board. The third board, the 8 port serial I/O board, is the network's interface to all the devices. The initial system will contain two of these boards. The final board is the interrupt board. This board chooses the highest priority interrupt which will be passed to the CPU board and also generates the interrupt signal.

3.1 CPU Board

The CPU board is the brain of the star network. A Z80 microprocessor is used in this design, because the 4MHz version of the Z-80 may possibly be used in a future version of the star network. Speed is a very important factor in this system. This 4 MHz version cannot be used until a faster program memory is used. There are two 2708, 1K EPROMs on this board that hold the system's program. These memories have an access time
of 500 ns, which is half the speed of the fast Z80. The 1K of RAM located on this card, is mainly used for the stack and for some temporary variables. This RAM has an access time of around 100 ns which is more than fast enough for a 4 MHz system. Finally, there is an Intel 8251 USART to do the serial I/O for a system monitor. This device uses the I/O request line rather than the memory request line, so the Z80 will add an extra wait state for this device. Thus, only a faster ROM is needed to increase the speed of the CPU card (See also Chapter V).

The diagram for the CPU board can be seen in the next five figures. Figure 3.1 illustrates the data bus structure of the system. The internal data bus structure consists of two unidirectional busses and are used for only the lower 32K locations. The DO# lines carry information from the processor and the DI# lines carry data to the processor. The DO# lines are always enabled, but the DI# lines are enabled when the \texttt{INT DATA IN EN} line is low. The external data bus, on the other hand, is a bidirectional bus. All external peripherals will have address locations in the upper 32K of memory. The \texttt{EXT DATA IN EN} signal enables data to flow from an external device to the processor. The \texttt{EXT DATA OUT EN} signal enables information to flow from the CPU board to another board. The external bus uses tri-state devices in both directions so that a possible second processor can use the system bus, while the main processor uses its internal bus.
Figure 3.1. CPU BOARD—DATA BUS STRUCTURE
Figure 3.2 shows the address bus and some of the processor generated control lines. These lines are unidirectional so the internal lines are always enabled. The external lines have tri-state devices because other processors must control these lines, too. The EXT BUS EN line controls the tri-stating of the address lines. The external RD and WR lines are open collector to keep these lines in a high impedance state. There should be no glitches at all on these two lines because data may be destroyed on the memory card or on the I/O cards if a glitch occurs. The M1 line is used as a clock on the Interrupt Board so it is always enabled. The Bus Req line is not used in this system but it will be used if a second processor is added.

Figure 3.3 illustrates most of the additional control logic on the processor board and it can be broken up into 4 smaller circuits. The first circuit is the reset generation circuit. A long reset pulse is needed by the CPU and other devices in the system to ensure proper initialization. The 74123, monostable chip generates this pulse on power up and also when manually reset. Below this monostable is the system clock generator. This circuit generates the 2 MHz clock signal that is used by the CPU and other devices. The crystal is easily changed to produce other clocking rates if desired. The wait line control and external bus control circuit are at the bottom of this figure. This circuit determines whether an off board device is requested and also determines whether an internal
Figure 3.2. CPU BOARD--ADDRESS BUS AND CONTROL
Figure 3.3. CPU BOARD--CONTROLLER
wait or external wait is requested. The wait line is not connected to the processor in the 2MHz system because no wait states are needed. Also the \text{BUS EN} input is strapped to ground since only one processor is being used and it will not have to share the buses. The final circuit on this drawing is the bus controlling circuit. Extended read, XRD, and extended write, XWR, pulses are generated to make sure that the buses are not disabled until after the RD or WR pulses go high. The rest of this circuit controls which bus should be enabled.

Figure 3.4 contains the serial I/O circuit and the ROM circuit. The Intel 8251 USART is the serial to parallel and parallel to serial converter. This is the only chip on the CPU board that has a bidirectional data bus, so the internal data buses are merged with a tristate buffer. This tristate is enabled during an internal write. Some special signals are generated, because the 8251 is designed for an 8080 system and not a Z-80 system. The delayed read, DRD, and delayed write, DWR, are needed because the read and write signals for the 8251 must occur after the chip is selected. The extended I/O request signal is used because the USART must still be selected after the write line goes high. The I/O circuit also contains a programmable baud rate generator to produce an accurate clocking rate for the input and output of the 8251. This generator also produces the internal clock for the USART. Finally, the 1488 and 1489 interface chips convert the TTL voltage levels to the voltage level of RS232. This serial I/O
Figure 3.4. CPU BOARD--SERIAL I/O AND ROM
port is accessed by doing an INPUT 1 or an OUTPUT 1 instruction. The status of this port is accessed by doing an INPUT 0 or OUTPUT 0 instruction.

There are two identical ROM circuits on this CPU board. These circuits use a 2708 EPROM for a memory element. Eventually a ROM with a 250 ns access time should replace these EPROM's so that the CPU can operate at 4 MHz. Each ROM uses a 7485 as an address decoder. The possible address locations of these ROMs are between 0000 and 3FFF. Initially these ROMs will use locations 0000 to 03FF and locations 0400 to 07FF.

Figure 3.5 illustrates the 1K by 8 RAM circuit which contains eight Intel 2125 memories. Since these RAMs have an access time of 100 ns, they do not need to be replaced when the system is modified. The RAM circuit uses the same address decoder as the ROM circuit. Initially the location of this memory will be from 3C00 to 3FFF.

3.2 Memory Board

This system has many large buffers and tables and an 8K RAM board is the correct size for a 24 port system. This RAM board is separated into two 4K circuits as pictured in Figures 3.6 and 3.7. Each circuit has its own address decoder, so that either group of 4K can be placed at any address in the system. Initially they will be located from addresses D000 to EFFF. This board also breaks up the system's bidirectional data bus into two internal unidirectional data
Figure 3.5. CPU BOARD--RAM
Figure 3.6. MEMORY BOARD—CONTROLLER
Figure 3.7. MEMORY BOARD--4K SECTIONS
buses, since the Intel 2125 memory chips do not have bidirectional data lines. The second of the figures shows how the control and data lines are connected to the memories. The address lines connect to all memory chips as described in the note on the drawing.

3.3 Serial I/O Board

The serial I/O boards interface the series I/O devices to the processor of the star network. Each I/O board contains eight identical serial ports and one controller for these ports. Figures 3.8 and 3.9 illustrate the controller of the I/O board and other common circuits. The address decoder, part of the I/O controller, is quite different from the decoders on the other boards. Three pins on the edge connector, the B.A. pins, determine what port numbers are on that I/O board. If pins 59, 17, and 60 are all low, the board contains ports 0 through 7. If pins 59 and 17 are low and pin 60 is high, the board contains ports 8 through 15. Thus each identical I/O board will have a different address because they will be plugged into a different coded connector. The boards must be placed in consecutive order with the first board in slot 0.

Once the port numbers are determined, the address locations that the board matches on are determined. Locations FF00 through FF3F are reserved for the status information of the Intel 8251 USARTs. If one reads location FF00, he will read the status of port 0. Port 1 will have its status location at FF91 and so on. Locations FF40 through FF7F are saved
Figure 3.8. I/O BOARD—CONTROLLER AND CLOCK
Figure 3.9. I/O BOARD—DATA BUS AND INTERRUPT BUFFERS
for the programmable baud rate of the ports. Again location FF40 corresponds to the baud rate of port 0. Finally, addresses FF80 through FFBF are the locations of the data in the USART. If one reads location FF84, he will read the input of port 4.

The I/O controller in Figure 3.8 has two major functions, address decoding and data bus control. The output of the 7485 comparator in the address decoder determines whether the board is being accessed or not. The output of the top 74155 multiplexer enables the proper USART on a status or data request. The lower 74155 enables the proper baud rate circuit on a baud rate request. The read enable signal, RD EN, and the write enable signal, WR EN, control the data buses on the I/O board. Data will flow into this board when the write enable signal is low.

Three small circuits are illustrated in Figures 3.8 and 3.9. The Baud Rate Clock Generator is a 5.07 MHz oscillator that is divided down by the baud rate generators in Figure 2.10 to create the proper clocking rates. This circuit has two outputs, CLK and CLK. The Data Bus Controller circuit is the second circuit. This circuit controls the bidirectional data bus that connects the USARTs to the processor. This circuit also controls the data bus that is connected to the baud rate generators. The last of these circuits is the interrupt line buffers. The USARTs do not have the capability to drive a 110 ohm line, so these buffers are needed. Also, these buffers isolate the USART from the system bus.
Figure 3.10 shows one of the eight identical I/O sections. The 74157 data selector controls the data path to the 7475 baud rate latch. On a reset, the switched baud rate is clocked into the baud rate latch. The data bus is connected to this latch when a new baud rate is written from the CPU. The 8T97, tristate buffer, is enabled when the baud rate is read. The programmable baud rate generator produces the internal clock for the USART and also produces the baud rate clock. The USART is the main component of the I/O section. The USART is connected to the CPU by the data bus and by some control lines. The data and handshaking signals are converted to RS232 levels in the 1488 and 1489 interface gates. The RTS line signals the device that the USART is ready to receive data. The DSR line signals the USART that the device is ready to receive data. The +12V line should be strapped to the DSR line on the connector if these handshaking lines are ignored.

3.4 Interrupt Board

The final board in this system is the interrupt board and is pictured in Figure 3.11. The main element in this circuit is a 48 level priority decoder. The upper 24 lines are for receiver interrupts and the lower 24 lines are for transmitter interrupts. The first six 74148 priority decoders determine which interrupt on the 48 lines has the highest priority and will put out the lower 3 bits of the number of the device.
Figure 3.10. I/O BOARD--ONE I/O SECTION
Figure 3.11. INTERRUPT BOARD
The seventh 74148 determines the middle 3 bits of the device number and also puts out the interrupt signal if there is an interrupt. The highest order bit is set if a transmitter is the interrupting device; if this bit is not set then a receiver is the interrupting device. The \( \overline{MI} \) signal from the CPU clocks the interrupting device number into a latch so that it is stable during a read. The address location of this latch is FFFF. If there are no interrupts, the \( \overline{INT} \) line will be high and a 3F will appear on the latch. An interrupt is removed when the interrupting device is read from or written to.

3.5 Mother Board and the System Busses

The backplane of this system is designed to minimize wave reflections at the ends of the lines and to minimize crosstalk between lines. This backplane consists of a one-sided mother board mounted over a brass ground plane. The mother board connects some of the pins of the board connectors together in a bus system, while other pins are not connected so that they can be used for special interboard signals. The lines in the bus system form a group of coupled transmission lines since high frequency signals appear on them. This section will briefly explain the design and construction of this bus.

There are many factors one must consider when designing the transmission lines in the bus. The characteristic impedance of the lines should be low to reduce the crosstalk between the lines, but it cannot be too low because the buffers cannot drive
low impedance lines. The dielectric constant of the mother-
board and the line width on this board determine the charac-
teristic impedance of the transmission lines, as shown in
the Wheeler Curves in Figure 3.12. The dielectric of the
motherboard should have good characteristics at high frequencies
to reduce the loss in these lines. Teflon or epoxy glass are
two examples of a good dielectric. A thick line will decrease
the ohmic loss of the line especially if the line width is
narrow. The line separation and line length also is an
important factor in the bus design. The lines should not be
too close together, otherwise the crosstalk between adjacent
tones will be high. The bus should be as short as possible
to reduce crosstalk.

Once the board is etched, the characteristic impedance
of the lines should be measured. A Time Domain Reflectometer
(TDR) is used to measure the reflection coefficient, \( p \), at
every point on the transmission line. The following equation
converts \( p \) to an impedance:

\[
Z = 50 \left( \frac{1 + \frac{P}{P}}{1 - \frac{P}{P}} \right).
\]

The characteristic impedance of the TDR is 50 ohms so a 50 ohm
cable must connect the TDR to the motherboard. The wire connec-
ting the cable to the board must be as short as possible to get
an accurate reading. Once the impedance is calculated, a
resistor of that impedance should be placed at the other end
of the line. The reflection coefficient of the line and the
reflection coefficient of the resistor should be the same on the TDR.

Shorting capacitors provide an AC short for the termination network at high frequencies, and allow a DC bias voltage to be placed on the bus. These capacitors should have a very low impedance between the systems clocking frequency and the fifth harmonic of this frequency. A Vector Impedance Meter (VIM) can measure the impedance of capacitors at these frequencies. The leads of the capacitor should be as short as possible to take these measurements. The best capacitors to use for this short are 360 microfarad @ 6V, Tantalex capacitors. These capacitors have an impedance less than 10 ohms below 20 MHz.

Figure 3.13 illustrates the electrical design of the system bus to minimize crosstalk and reflections. There are two active terminations to put a 2.6 volt bias voltage on the bus. This will put all inputs to a high state but near the switching level to reduce crosstalk. The 360 microfarad capacitors provide the AC ground that terminates the data bus. 180 ohm resistors, rather than 100 ohm resistors, terminate one side of the data bus, because the drivers cannot sink enough current from the termination network to produce a reliable low voltage level with the 100 ohm resistors. There will be a reflection coefficient of .3 at this end of the bus because it is not terminated in 100 ohms. The data lines are positioned a few inches away from the address lines to reduce crosstalk between the two groups. Spacing is not as critical
Figure 3.13. SYSTEM BUS
within each group since all lines change states at the same time. Close control lines like the RDM and WRM lines, are separated by a ground line to reduce crosstalk between them. These lines are terminated at both ends to cut down on crosstalk even more and also to reduce reflections.

This is just a first attempt to design a high speed bus system. If higher speeds are needed, the crosstalk and line reflections should be reduced even more. The line impedance could be reduced from 100 ohms to reduce the crosstalk problem. This is done by adding a second ground plane to the motherboard on the other side of the transmission lines. Plated holes in the board would make this bus easier to build. The line reflections could be greatly reduced if both ends of all the lines are terminated. The second termination can be 100 ohms or lower because no DC current flows through this resistor to upset the buffers. Capacitors should be placed from each line to ground unlike the termination on the other end, so that no DC current flows from one line to another. The value of these capacitors should be less than 500 pf, because this RC network must have a fast rise time.
CHAPTER IV
SOFTWARE

The discussion of the software in this system is broken up into two parts. First, the four major divisions of the system's program are discussed. This will just be a brief description of what each part does. The second part of this chapter describes how the system's tables and buffers are arranged and it will be a more detailed description of specific parts of the program.

4.1 The System's Program

The system's program is broken up into four major divisions, and the first one is the initialization routine. This routine executes immediately after a reset occurs. First, the number of 8 port I/O boards in this system are determined. Then all the tables and pointers are initialized. The USARTs are then initialized into the asynchronous mode. Finally, the USARTs are read twice to remove bad data.

After the system is initialized, the enable loop routine is entered. The system allows six interrupts to occur before the next transmitter and receiver are scanned. This scan checks to see if either is disabled. The transmitter will be enabled if its buffer is not empty and the device is ready to receive data. The receiver will be enabled if its buffer
is not full. After this scan is done, the port number is incremented and then the enable loop is reentered.

Figure 4.1 contains the flowchart of the interrupt routine. First this routine reads the device number and then determines if this is a transmitter interrupt or a receiver interrupt. If this is a transmitter interrupt, then the status of the buffer is checked. The transmitter will be disabled if its buffer is empty. Otherwise the next data word is transmitted to the device. If this is a receiver interrupt, then the following sequence of events occurs.

The incoming data must be a valid character, otherwise the program jumps to the error section of the command routine. Next, the interrupt routine determines if the escape sequence was entered, and will jump to the escape section of the command routine if it was entered. If the escape sequence was not entered, the data is placed in a buffer. If the buffer becomes full then this receiver's interrupt is disabled and the device is signaled not to send any more data.

Figure 4.2 contains the flowchart of the command routine which can be entered in two places. The first entry point is in the error section of this routine. If the port is in command mode, then the error is just ignored so it will not be echoed back to the device. If the port is in connect mode, then all devices connected to this port are placed into command mode, and an error message will be sent to each of these devices.
Figure 4.1. INTERRUPT ROUTINE
Figure 4.2. COMMAND ROUTINE
The second entry point is in the escape section of this routine. If the escape sequence is entered and if the device is in connect mode, then all devices connected to this port are placed in command mode and are sent the prompt sequence. If the device is in command mode, then the processor decodes the command in the receiver buffer. The command will be executed if it is valid. Otherwise the command is ignored, and an error message will be sent to the terminal.

4.2 The System's Tables and Buffers

The processor uses the tables and buffers for three major functions. The processor uses the three tables in Figure 4.3 to control the data flow through the system. The Receiver Buffer Table (RBT) contains the location of the buffer that stores the data from a given port. This is a table of two byte entries. The port number must be multiplied by two and added to the address D000 to find the position of the low order byte. The high order byte is in the next location in the table. The Transmitter Buffer Table (TBT) holds the buffer location that contains the data to be transmitted for a given port. This table is accessed the same way that the RBT is, except D080 is added to twice the port number. If both tables contain the same buffer position for a given port, then the port is in command mode. The buffer is a 256 word circular buffer, and it contains four pointers. Two of these pointers, the begin pointer (BGN PTR) and the end pointer (END PTR)
Figure 4.3. TABLES FOR BUFFER CONTROL
contain the address of the first and last position in the buffer. The receive pointer (REC PTR) contains the location in the buffer to store the next data byte. This pointer is updated after a new byte is put into the buffer. If this pointer becomes equal to the transmit pointer after insertion, then the buffer has just filled. The transmit pointer (XMT PTR) contains the buffer location to retrieve the next data byte. This pointer is updated when a data byte is taken from the buffer. If this pointer becomes equal to the receive pointer after deletion, then the buffer is empty.

The command table and the buffers hold data which the processor uses to control the command recognition routine. The command table contains the location of the commands in each buffer. The position of an entry in this table is found by adding the port (or buffer) number to EB@. The receive pointer of the buffer contains the location that is one byte beyond the last character of the command. Two examples of this process are pictured in Figure 4.4. The command in buffer 0 starts at location 06. The command ends at location 0A which is one position before the contents of the receive pointer. The command in buffer 1 begins at location 09. The end of the command is also position 09 which is one position before the contents of the receive pointer.

The processor uses the escape table to determine if a command has been terminated. The structure of this table is illustrated in Figure 4.5. Locations EC00 to EC17 of this
Figure 4.4. TABLES FOR COMMAND RECOGNITION
Figure 4.5. TABLE FOR ESCAPE SEQUENCE CONTROL
The table are pointers to each port's escape sequence. The pointer of a given port is found at a location of EC00 added to the port number. If this pointer is a 00, then the port does not have an escape sequence. This port cannot execute commands or terminate a connection. If the pointer is a 38, then the port is in command mode and a carriage return is the escape sequence. If the pointer is a different value, then the port is in connect mode, and it uses its own escape sequence. The lower part of the escape table contains a group of 25 lists that hold the escape sequence for each port. These lists start at location EC38 and end at ECFF. Each list is allocated 8 bytes and contains two pointers. Next pointer (NXT PTR) holds the location of the next character to be entered in the escape sequence. End pointer (END PTR) contains the location of the last character in the sequence. The list starting at location EC38 is used for ports that are in command mode and that can execute commands. The list starting at position EC40 contains the escape sequence for port 0 when it is in connect mode. The list for port 1 would start location EC48 and so on.
CHAPTER V

FUTURE MODIFICATIONS

This star network has been designed with the purpose to show that a microprocessor system can control a large number of devices. The present system can transfer approximately 3,000 data bytes per second. This rate can be increased by 10% if only one port in every connection is allowed to terminate the connection. This rate may not be fast enough if the system is used extensively. This chapter explains how to modify the system to increase the transfer rate.

5.1 A Minor Software Modification

The data transfer rate of this system can be increased if two software changes are made. The first and last memory position of a buffer is presently looked up because the length of the buffers can be variable. If the buffers are of a fixed length, then the Receiver Buffer Table, and the Begin Pointer and the End Pointer of each buffer, can be eliminated because these values can be calculated. This calculation is much faster than using a lookup table. The Transmit Buffer Table uses two byte entries to locate the proper table because the buffers can start at any memory position. If all buffers start at a memory position with $00$ as the lower byte address, then
this table can be reduced to one entry per buffer. If these two software changes are made, then the system's transfer rate can be increased by 15%.

5.2 A Minor Hardware Modification

The hardware of this system was originally designed with a Z80-A microprocessor, which is the 4 MHz version of the Z80 microprocessor. The 4 MHz version is not used in the final design, because a 2 MHz ROM is the fastest ROM available (within a reasonable price range) for use as program memory. If a faster program memory is used, the following changes must be made. First, a Z80-A processor should replace the Z80, because the Z80 is operating at its highest speed. Also, the ROM sockets may need to be rewired to accept the new ROMs. Next, the wait line must be connected to the processor. There are still some 2 MHz devices on this system so that one wait state is added when accessing these devices. The USARTs on the I/O board are 2 MHz devices.

If fast ROMs cannot be found, then the following trick can be used. There is 1K of 100 ns RAM on this board. If 1K more of this RAM is put onto this board, then the program can be moved from ROM to RAM. The processor can operate at 4 MHz, because now the RAM contains this program. The data transfer rate will almost double if a faster program memory and CPU is used.
5.3 A Major System Modification

The speed of the system may be very inadequate if the system is expanded. The system will have more processing power to increase this speed, if a board with four parallel processors replaces the single CPU board. This change will cause the interrupt board to be replaced by a board with a microprocessor that controls the system. The four parallel processors will be slaves to this control processor. The system's software must be separated into groups because the two types of processors have two different functions. This change is discussed in this section of the paper.

The structure of these four slave processors should be similar to the structure of the single processor. Each of the slaves should have its own internal bus system to access 2K of ROM and to access a small amount of RAM. A USART is not needed for these processors. Each CPU should have its own bidirectional tristate buffer to access the system bus. This entire structure is important because each of the slaves will be independent of the others for its program memory. Each slave will use the system data bus for 4% of the cycles in the interrupt routine, so there should be little contention for this bus.

The interrupt board must be redesigned to perform the functions in Figure 5.1, by expanding the existing CPU board. A group of parallel input ports are added so that this processor can scan the interrupt lines and choose the next device to be
Figure 5.1. FUTURE INTERRUPT BOARD
handled. The CPU will then put this device number into the Interrupt Latch which is still read at location FFFF on the system bus. This CPU must control the other four processors, hence the External Processor Controller (EPC) function is added to this card. The CPU scans the four halt lines to see which of the slaves is idle. The controller uses the interrupt request lines to get a slave to respond to an interrupting device. The interrupt acknowledge lines signal the EPC to clear the interrupt request line. A final function that is added to this board is the System Bus Controller. The Bus Request signal from the five processors, request the system data bus. This signal can be implemented by a 32 by 5 ROM. This circuit also generates a five phase clock which is used by all processors. This will cause the Bus Request signal of each processor to occur at a different time of a given cycle.

The four software routines of the one processor system, can be used in this new system, by making a few minor changes. Two of the routines, the interrupt routine and the command routine, are used as program memory for the slave processors. There are changes made to these routines, but when the processor returns from either routine, it must halt to signal the control processor that the slave is idle. The other two routines, the initialization routine and the enable loop, are used by the control processor as part of its program memory. The control processor will initialize the system after a reset, so the slaves should be halted. After the reset, the control processor should execute a scan routine which replaces the first part of
the enable loop of the existing system. This routine determines the next interrupting device and controls the EPC. The control processor then executes the second part of the old enable loop.