MICROCOMPUTER NETWORK

BY

PETER SHAW MOLDAUER

B.S., University of Illinois, 1979

THESIS

Submitted in partial fufillment 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, 1981

Urbana,Illinois

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to my thesis advisors, Prof. Ricardo Uribe and Prof. Michael Schlansker, for their guidance and assistance in this project. For inspiration during the conception of this project, I would like to thank Michael Pogue.

Special thanks to Jim Graf for finding software problems and to David Wachter, Steve Schmitt and Prof. Ricardo Uribe for helping build the microcomputer network.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The microcomputer network is a powerful processing machine that is constructed using a number of small microcomputers. ach microcomputer is based on a single chip microcomputer supplemented with memory and peripherals. The network was designed to be a general purpose machine to model the interactions of several programs running asynchronously. The requirements of such a model are easily met by the microcomputer network, sufficient memory space is available, the basic execution rate is variable, and a random number generator is included. The microcomputer network is general enough for a wide range of applications, therefore a section on programming the network is included.

The next chapter is an overview of the network. Chapter three covers the design and construction of the network hardware. The software that runs on the network is discussed in chapter four along with a section on programming the network. A list of references is included for specific information on devices used in the design.

CHAPTER 2

SYSTEM OVERVIEW

Since the microcomputer network is a general purpose processing machine, the hardware is capable of supporting many diverse applications. The software in the system allows access to all of the major attributes of the network. The network is constructed such that each functional unit in the network is physically located on an individual board or card. The network is a collection of three different types of cards; the arbitrator microcomputer, the node microcomputer, and the terminal interface board. A typical configuration includes one arbitrator microcomputer, one terminal interface board, and several node microcomputer boards. The processing done by the network is performed in the node microcomputer boards. The node microcomputers are completely independent, there is no shared memory in the system. Each node microcomputer is capable of executing any segment of code independently of the rest of the

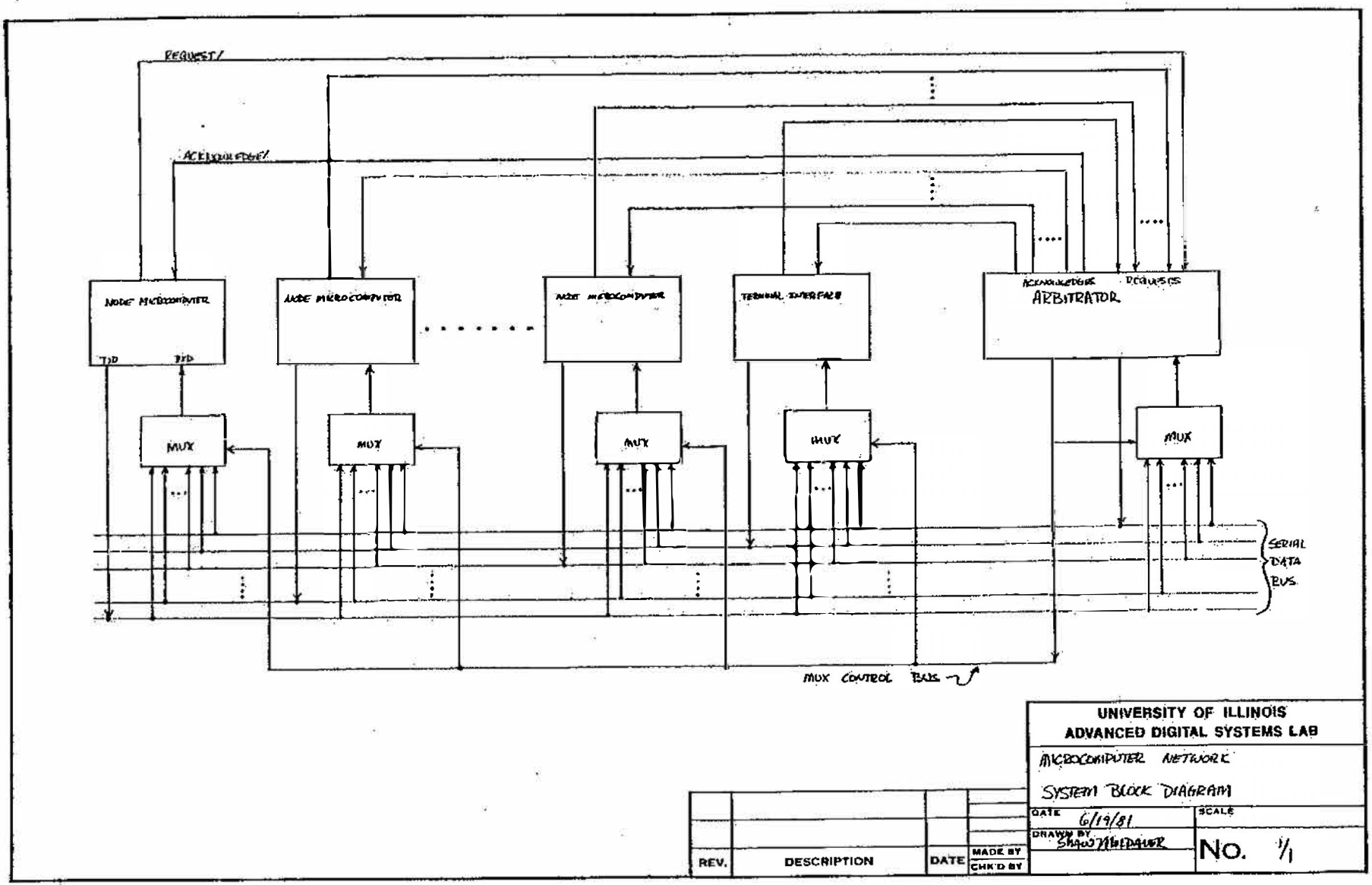
network. Communications between nodes in the network is done serially. A maximum of sixteen active boards are supported by the system allowing a system of fourteen nodes, one terminal interface and an arbitrator.

The arbitrator microcomputer is responsible for handling all of the network reconfiguration capability. All node microcomputer boards and terminal interface boards are connected through the backplane with the arbitrator microcomputer. The arbitrator microcomputer is not available to the system user for programming, as it does not have any means of accepting or transmitting information to or from the user. The software that the arbitrator processor executes is fixed in its memory and is invoked only through certain sequences of commands from a node processor or a terminal interface board.

Since the node microcomputer is responsible for all processing power in the network, it is the most generalized part of the network. The node microcomputer consists of a general purpose microcomputer supplemented with memory and input/output (I/O) devices. No direct access to the node processor exists in the network. All communication with the external world occurrs through a terminal interface board. Software on the node processor board includes a basic serial I/O based monitor supplemented with useful subroutines. The node processor is constructed in such a manner as to allow both hardware and software expansion.

The terminal interface board allows the external world to access the operations of the network. To the network, the terminal interface board appears to be a node processor board, but its sole purpose is to connect to a device that communicates serially.

A block diagram of the microcomputer network is shown in figure 2.1.1. The communications between nodes is done through the serial data bus. The arbitrator processor controls each node serial input multiplexer. Each node has a seperate request and acknowledge signal which provide handshaking with the arbitrator.



REV.	DESCRIPTION	DATE	MADE BY / CHK'D BY

UNIVERSITY OF ILLINOIS
ADVANCED DIGITAL SYSTEMS LAB

MICROCOMPUTER NETWORK

SYSTEM BLOCK DIAGRAM

DATE 6/19/81 | SCALE

DRAWN BY Shaoo Mehdiabadi | No. 1/1

FIG. 2.1.1 SYSTEM BLOCK DIAGRAM

CHAPTER 3

SYSTEM HARDWARE

The microcomputer network is a collection of three types of circuit boards; node microcomputers, terminal interfaces and one arbitrator. This chapter discusses the boards individually, and includes a discription of the rack, backplane and power supply. Each board is discussed briefly and this is followed by sections detailing the functional blocks associated with the board. Schematics of the logic design are included in the discription.

3.1 Node Microcomputer

The node microcomputers contain all the circuitry necessary to be complete microcomputer systems. Each node has a central processor that controls the actions of that node. This processor is an Intel 8748 single chip microcomputer. Additional circuitry

is provided to facilitate expansion of system memory, generation of random numbers, serial communications and system bus interface.

Provisions for extension of support hardware remain for possible applications involving interface with the external environment; only one of the two eight bit ports on the 8748 processor has been used in the design of the node microcomputer. In addition, several addressable locations have been decoded but remain undedicated.

3.1.1 Central Processor

The 8748 single chip microcomputer is the heart of the node microcomputer. Internal to the 8748 are many of the functional blocks necessary for microcomputer operation. The 8748 processor is a single chip microcomputer from the Intel 8048 family of microcomputers. The family includes the 8048 (with internal factory programmed read only memory (ROM)), the 8035 (with provisions for external erasable programmable read only memory (EPROM)), and the 8748 (with internal EPROM). The node microcomputer is shown in figure 3.1.1.

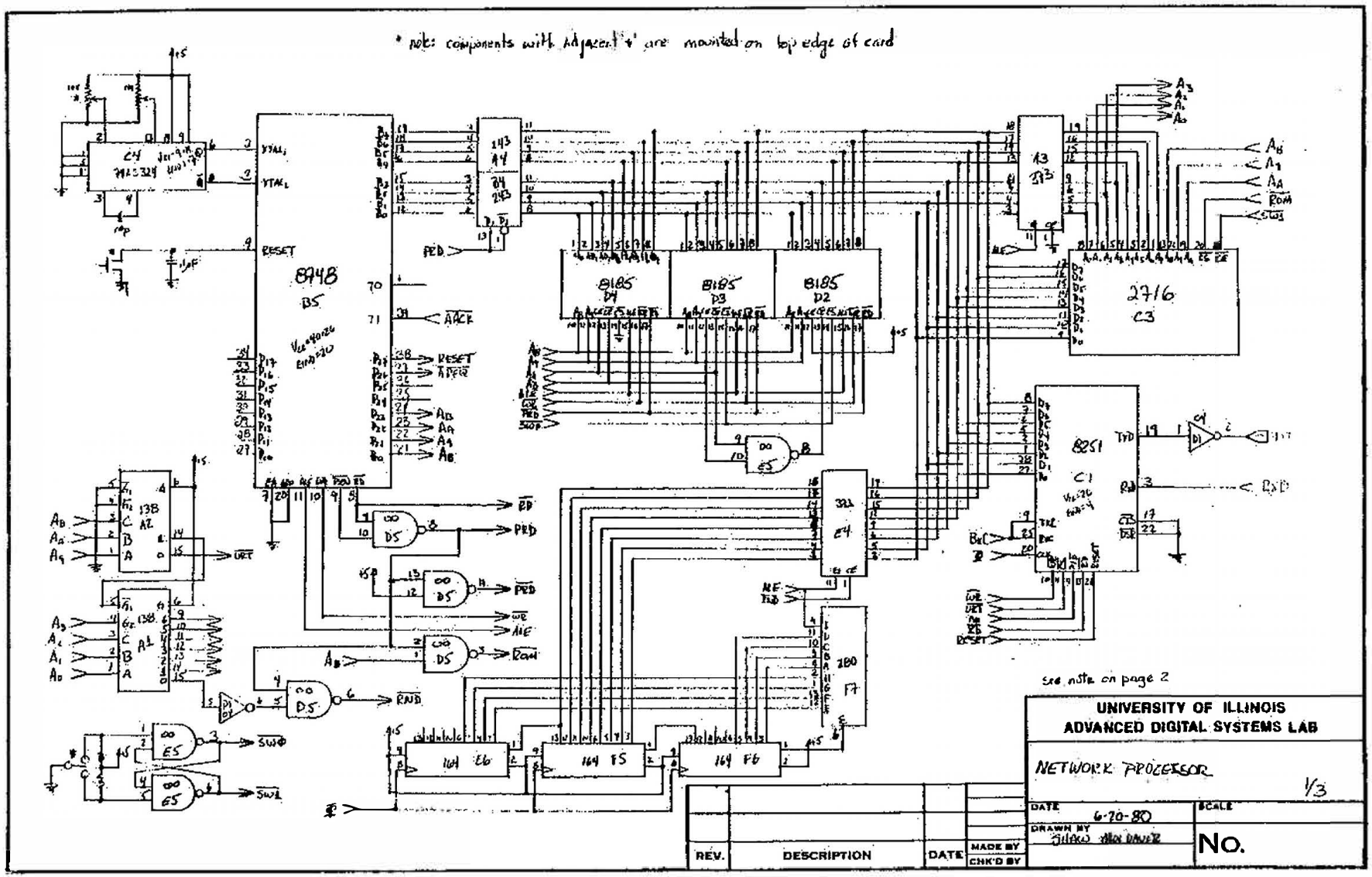


FIG. 3.1.1 NODE MICROCOMPUTER

The 8048 family of microcomputer have all of the circuits for simple applications internal to the processor. The center of the 8048 is the accumulator, an eight bit register which is the source or destination for most operations. An eight bit arithmetic logic unit performs the arithmetic and logical operations specified in the instruction stream. Both random access memory (RAM) and ROM have been designed into the 8048, there are 1024 bytes of ROM internal to the chip (except on 8035). The RAM that is in the chip is used to provide for all working registers, stack memory and user data memory. Two banks of working registers are allocated in the 64 byte RAM. Either of the two banks can be activated by software control. The processor stack is also allocated in the internal RAM. When subroutine calls or interrupts are processed, the return address and certain processor flags are stored in the stack. Up to eight levels of subroutines are provided in the internal RAM. A timer circuit is also built into the 8048 family to allow for counting or timing.

The last major functional block that is internal to the 8048 microcomputer is the multitude of I/O pins available to the user. Two eight bit ports can be used as inputs or outputs or combinations thereof. Two test inputs that can be selected as the condition code for a branch instruction are also available to the user. The BUS port, a true bi-directional data bus, is used for external reads, writes and instruction fetches. It may optionally be used as a static port if desired. When used as a bi-directional

data bus, the address is multiplexed out the BUS port immediately prior to the data transfer.

3.1.2 External Memory

In the design of the node microcomputer, the quantity of internal memory was found insufficent for a reasonable size application. Therefore, the memory was supplemented externally. A convenient method of storing programs is in EPROM. Provision for 2048 bytes of additional program memory was realized with a socket for a 2716 EPROM. Another fundamental shortcoming of the 8048 family is the lack of executable RAM. The node microcomputer has sockets for 3072 bytes of external RAM using three Intel 8185 RAMs. The largest executable address space of the 8048 family is 4096 bytes, or twelve bits of address specification. A switch has been placed on the node microcomputer board to allow the programmer to choose which type of memory occupies the top half of executable memory. The programmer may select between a system with 3K ROM and 1K RAM or a system with 1K ROM and 3K RAM.

3.1.3 Serial Interface

The serial communications interface is realized with an Intel 8251 universal synchonous asynchronous receiver transmitter (USART). The 8251 is capable of handling full duplex serial transactions independently of the 8748 microcomputer. The USART's address was not maximally decoded, allowing the processor to interact with the USART while destroying as few registers as possible. The output line of the USART is directed to the system bus for use by other node microcomputers. The input line of the USART is connected to any of the outputs of the other node microcomputers by the control of the arbitrator processor. This will be explained in the bus interface section.

3.1.4 Random Number Generator

A concept that was found useful in the design of application software was a random number generator. The choice between a software implementation and a hardware realization was studied. Either implementation of the random number generator was found to give a sequence generator, this means that the sequence of 'random' numbers would repeat after some finite sequence length; and each cycle through the sequence generates the same numbers in the same

order. The minimal instruction set of the 8048 family favored the hardware realization since the address space of the machine is small. In the hardware a 24 bit binary sum generator was chosen. An eight bit tap was taken from the internal state and used as the random number generator output. The circuit for this method was breadboarded and the length of the sequence tested. The basic circuit was found to have a sequence length of around 750,000 numbers. This circuit gives a reasonable sequence length, but when an asynchronous frequency is also added to the sum, the length of the sequence, if finite, becomes very long (over 5,000,000 numbers). The asynchronous frequency was realized by summing the processor clock (ALE) with the feedback from the shift register and clocking the register with the bus clock.

3.1.5 Processor Clock

The node microcomputer was designed without a crystal oscillator in order that the basic execution rate be variable. This concept allows the user to set each node to a different operating speed, and ensures that a long sequence is maintained in the random number generator. The variable frequency clock was realized with a voltage controlled oscillator; the frequency is then a function of the position of the clock rate potentiometer on the front of the node microcomputer card.

3.1.6 System Bus Interface

Each node has the circuitry necessary to initiate an interaction with any other node in the system, or the arbitrator. The node microcomputer can perform a request to have its serial port connected to the serial port on the arbitrator processor. When the arbitrator acknowledges this request, the node microcomputer may submit any valid interconection or status request to the arbitrator through the serial link. The arbitrator processor has complete control over every node microcomputers serial input connection and therefore can interconnect any node as the request specifies. The system bus interface controls the only output feature of the node microcomputer's front panel, a light emmiting diode. It allows the user to see each time a node has become in communication with the arbitrator.

In addition to the interconnection aspect of the system bus interface, the node microcomputers receive all the critical timing signals pertaining to the serial communications from the bus. The basic system bus clock is used by the node microcomputers to satisfy the USARTs requirement for a high speed clock and to clock the shift register used to generate the random numbers. The baud clock is used by the node microcomputers to synchronize the serial communications. The bus interface is shown in figures 3.1.2 and 3.1.3.

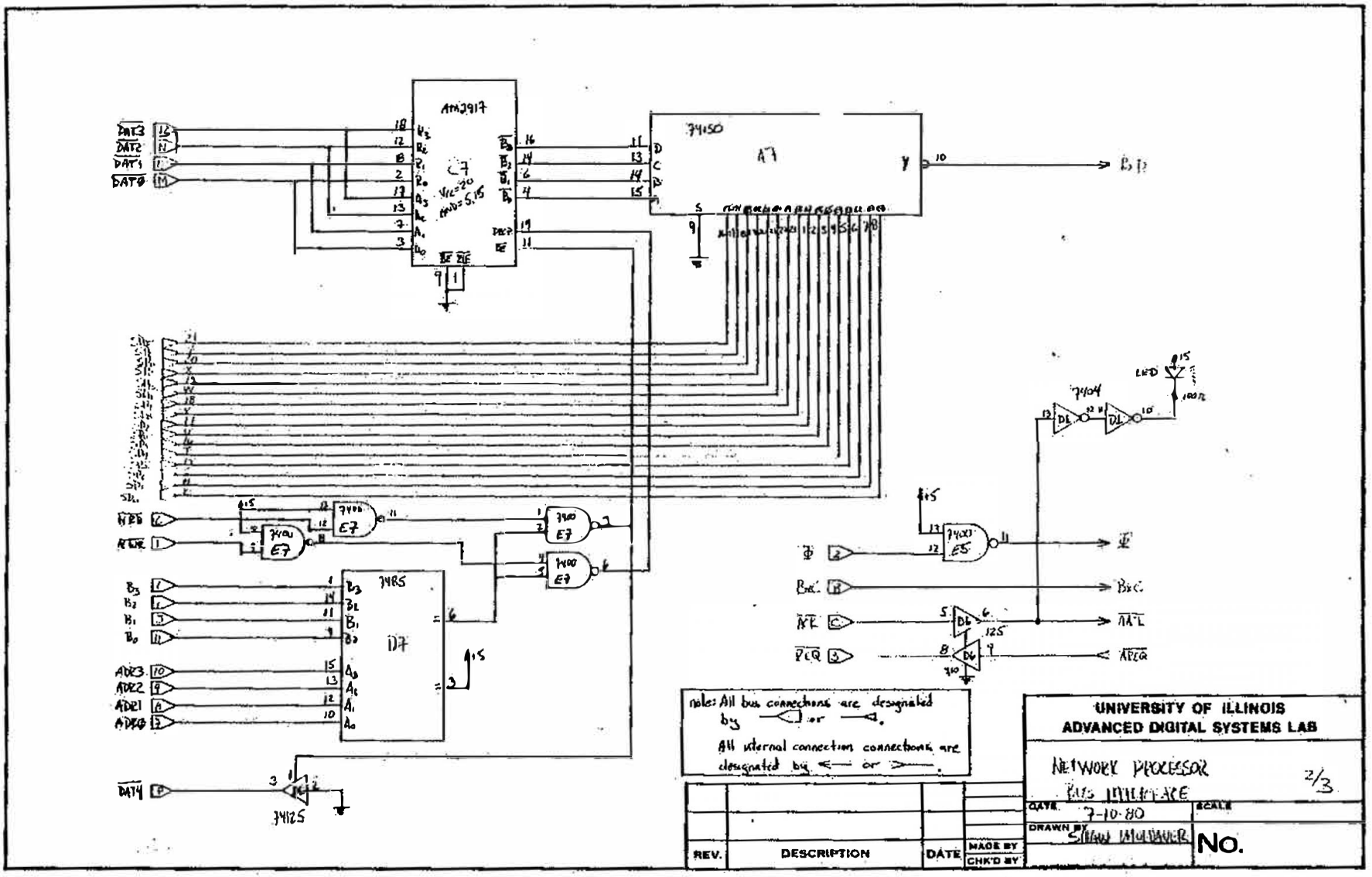


FIG. 3.1.2 NODE MICROCOMPUTER BUS INTERFACE

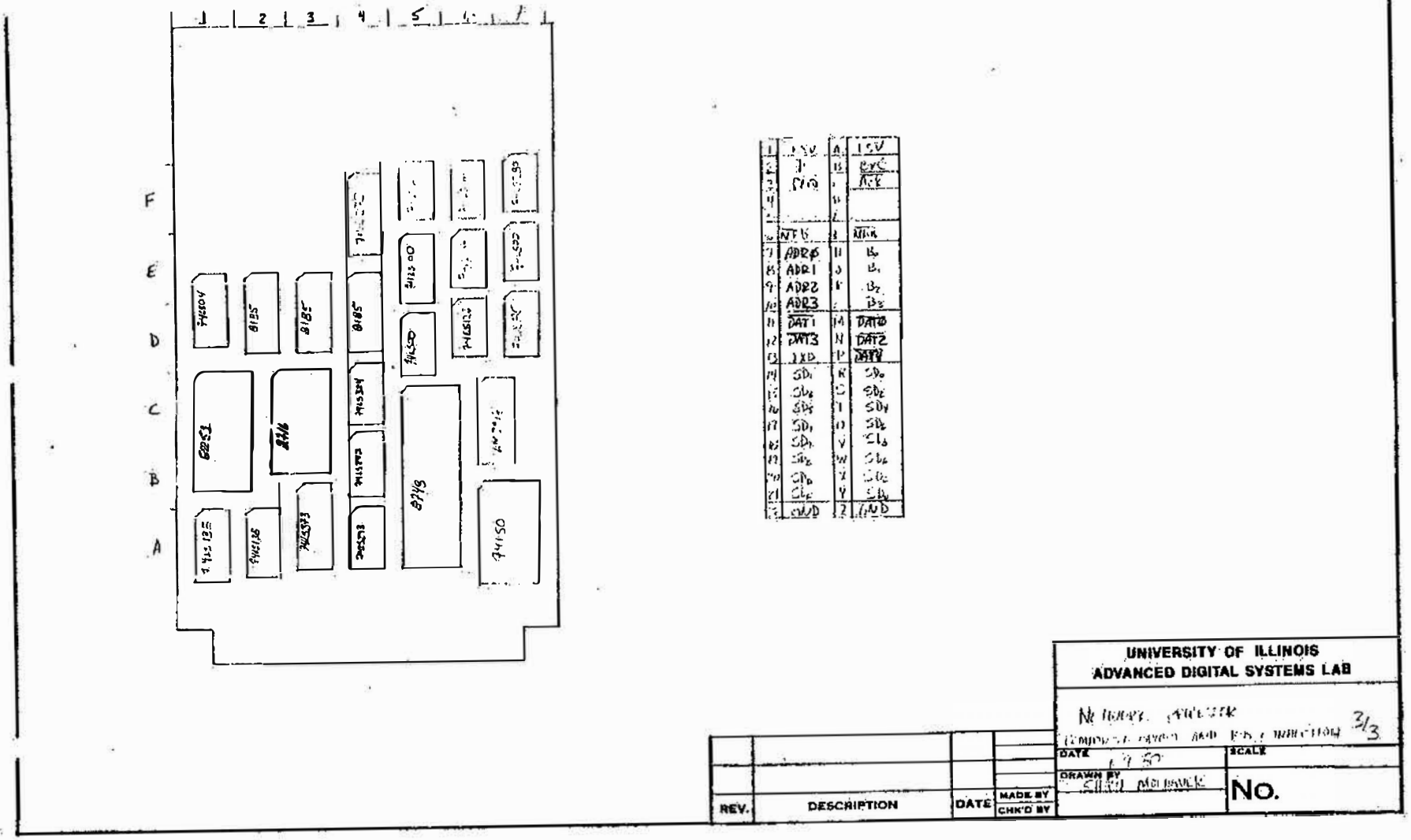


FIG. 3.1-3 NODE MICROCOMPUTER BOARD LAYOUT

3.2 Terminal Interface

The terminal interface board is a functional subset of the node microcomputer. It contains only the bus interface section of the node microcomputer. This terminal board allows a normal EIA compatible serial terminal to be connected to any of the processor boards in the multi processor network. The front panel is slightly different than that of the node, as there is no clock rate potentiometer or memory select switch. The acknowledge light is functionally equivalent to the one on the node microcomputer board. The pushbutton on the terminal interface board generates an arbitrator request, this allows the user to force access to the network to run a program on a node microcomputer or examine the status of the network. The terminal interface board is shown in figure 3.2.1.

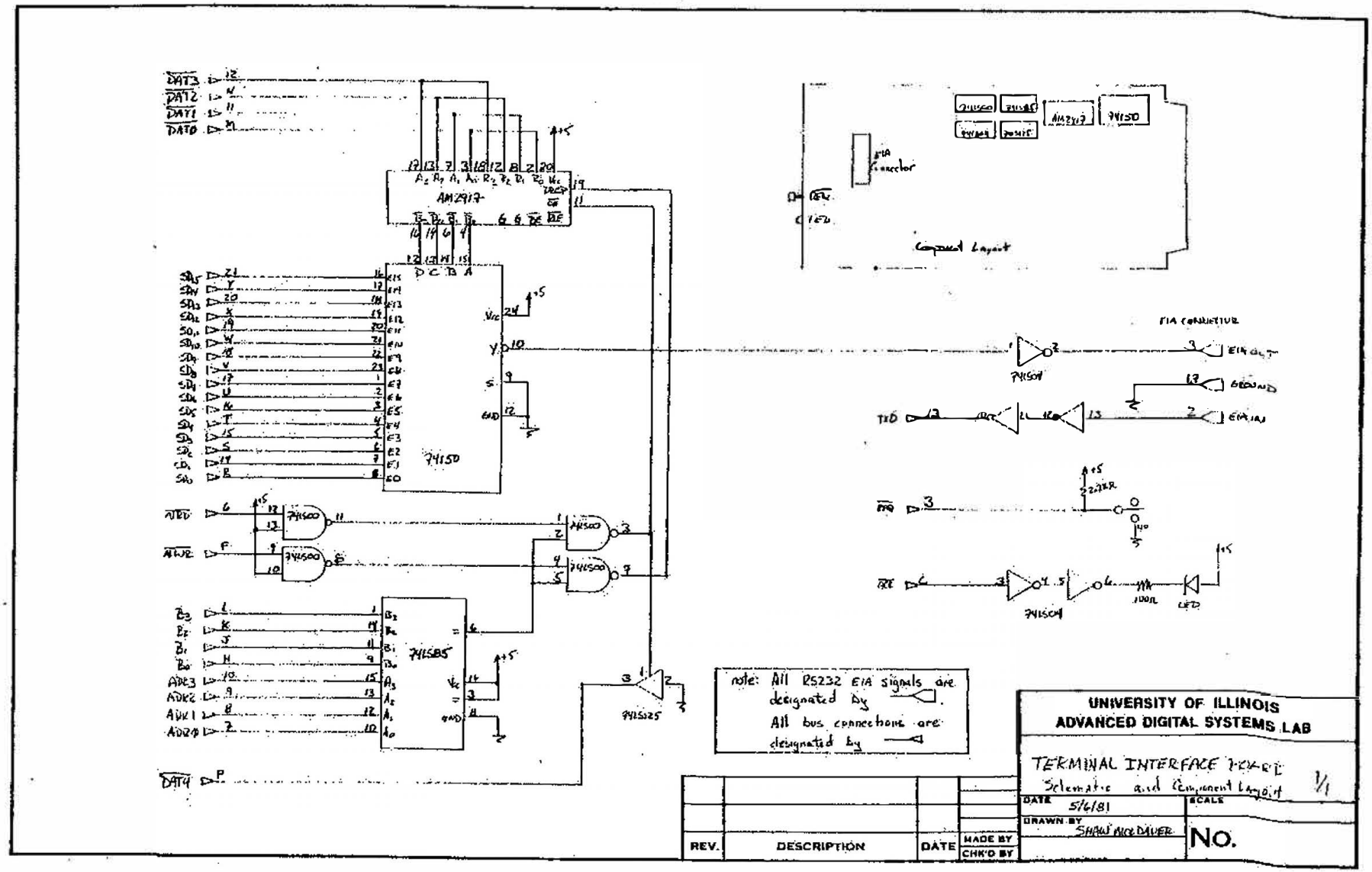


FIG. 3.2.1 TERMINAL INTERFACE BOARD

3.3 Arbitrator

The arbitrator processor is a self-contained microprocessor board based on Intel's 8085 eight bit microprocessor. The arbitrator generates all the critical timing signals for the entire multiprocessor system and also provides the arbitration and interconnection capability. this allows multiple simultaneous serial interactions between independent node microcomputers. The arbitrator processor has a defined set of posssible actions, and is therefore not available for system expansion. Also the physical card on which the arbitrator resides is densely populated.

3.3.1 Central Processor

The 8085 microprocessor which controls the arbitrator is an advanced version of the industry standard 8080 microprocessor. The instruction set of the chip is extensive and powerful. This processor chip was chosen for the arbitrator because of the ease of programming, speed of execution and simplicity of interface. The 8085 CPU has an eight bit external data path and addresses 65,536 bytes of memory. The 8085 also addresses up to 256 different input or output devices, although this feature is not used in the arbitrator. Control signals for the memory interface are generated

internally to the 8085 CPU. The arbitrator central processor is shown in figure 3.3.1.

3.3.2 External Memory

The single board arbitrator processor contains all the memory necessary for operation of the system. The program which the 8085 CPU executes is stored in a 2048 byte EPROM (Intel 2716), variable storage and stack storage are allocated in a 1024 byte RAM constructed of two 1024 by four bit RAMS (Intel 2142). The EPROM occupies locations 0000H through 07FFH and the RAM occupies locations 3C00H through 3FFFH. Either memory is available for processor reads or instruction fetches and the RAM is available for processor writes. The external memory is shown in figure 3.3.2 and the arbitrator address decoding is shown in figure 3.3.3.

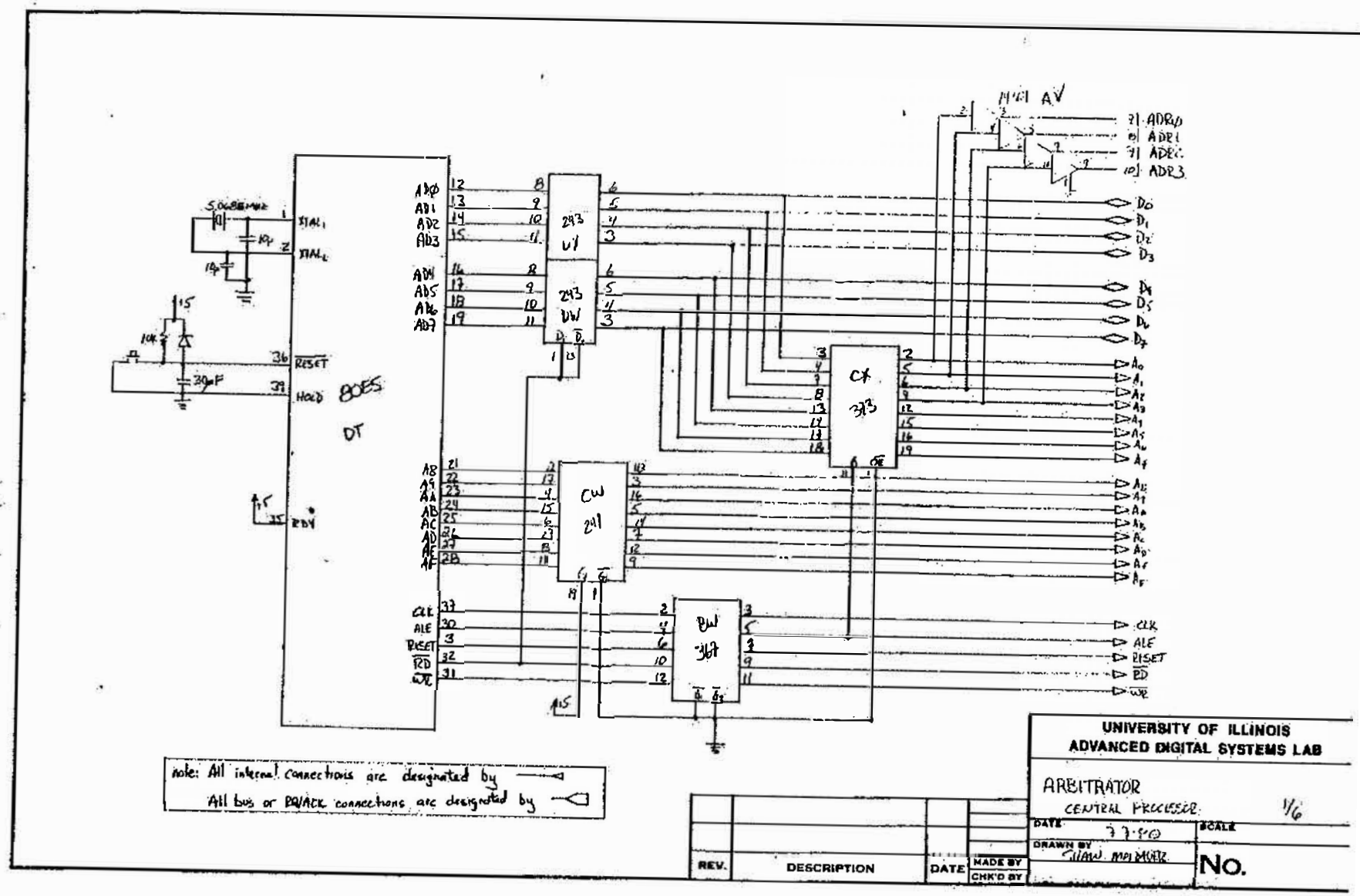


FIG. 3.3.1 ARBITRATOR CENTRAL PROCESSOR

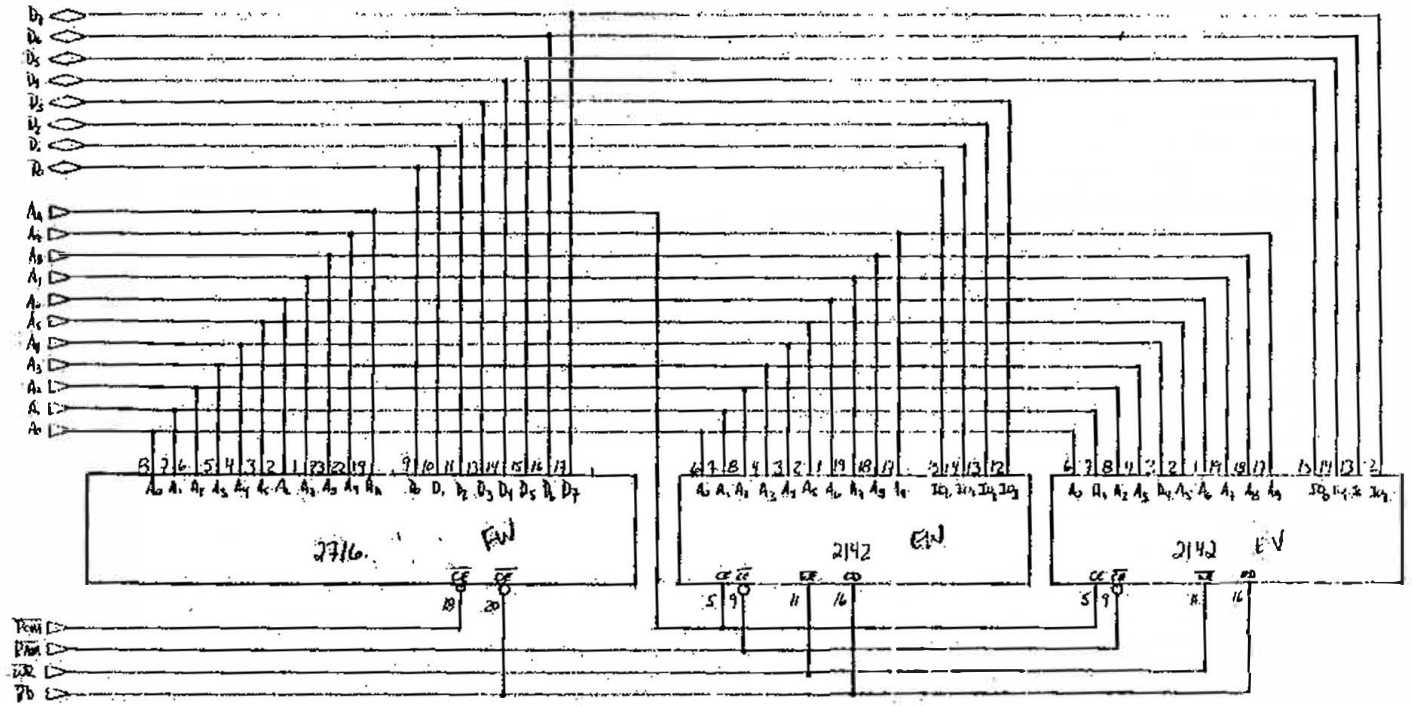


FIG. 3.3.2 ARBITRATOR MEMORY

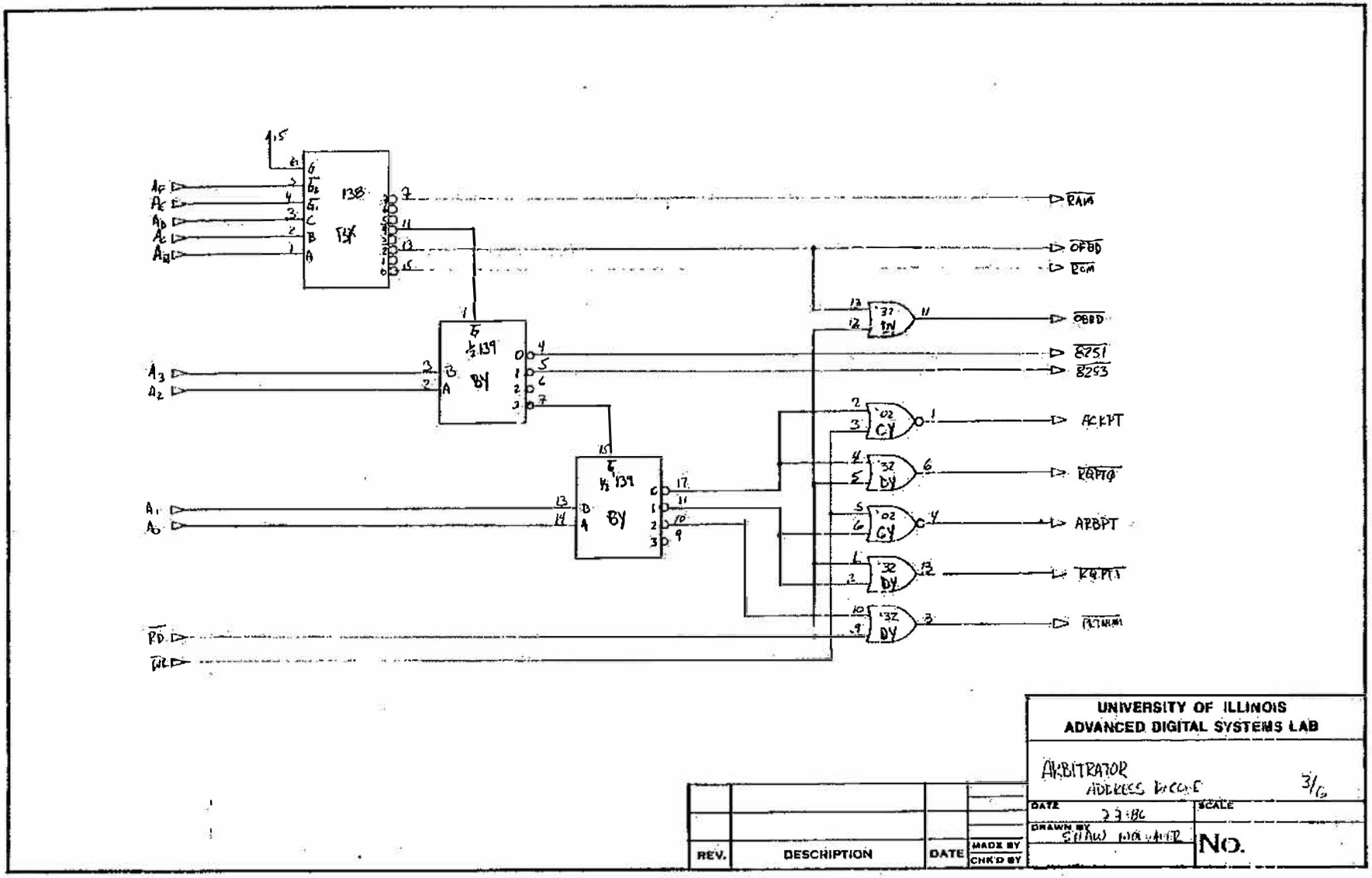


FIG. 3.3.3 ARBITRATOR ADDRESS DECODE

3.3.3 Serial Interface

The serial communications interface is realized with an Intel 8251 USART. The 8251 is capable of handling full duplex serial transactions independently of the 8085 processor. The USART occupies a memory address for simplicity of system address decoding. The physical address is decoded to be 2000H. The output line of the USART is directed to the system bus for use by the node microcomputers. The input line of the USART is connected to the output of any of the node microcomputers by a multiplexer on the arbitrator processor. A four bit I/O port on the arbitrator controls the input multiplexer. The bit rate (BAUD rate) is generated in an Intel 8253 programmable interval timer. This allows the BAUD rate to be dynamically changed. The arbitrator processor supplies the network with all the clocks necessary for the serial communications. The arbitrator serial I/O is shown in figure 3.3.4.

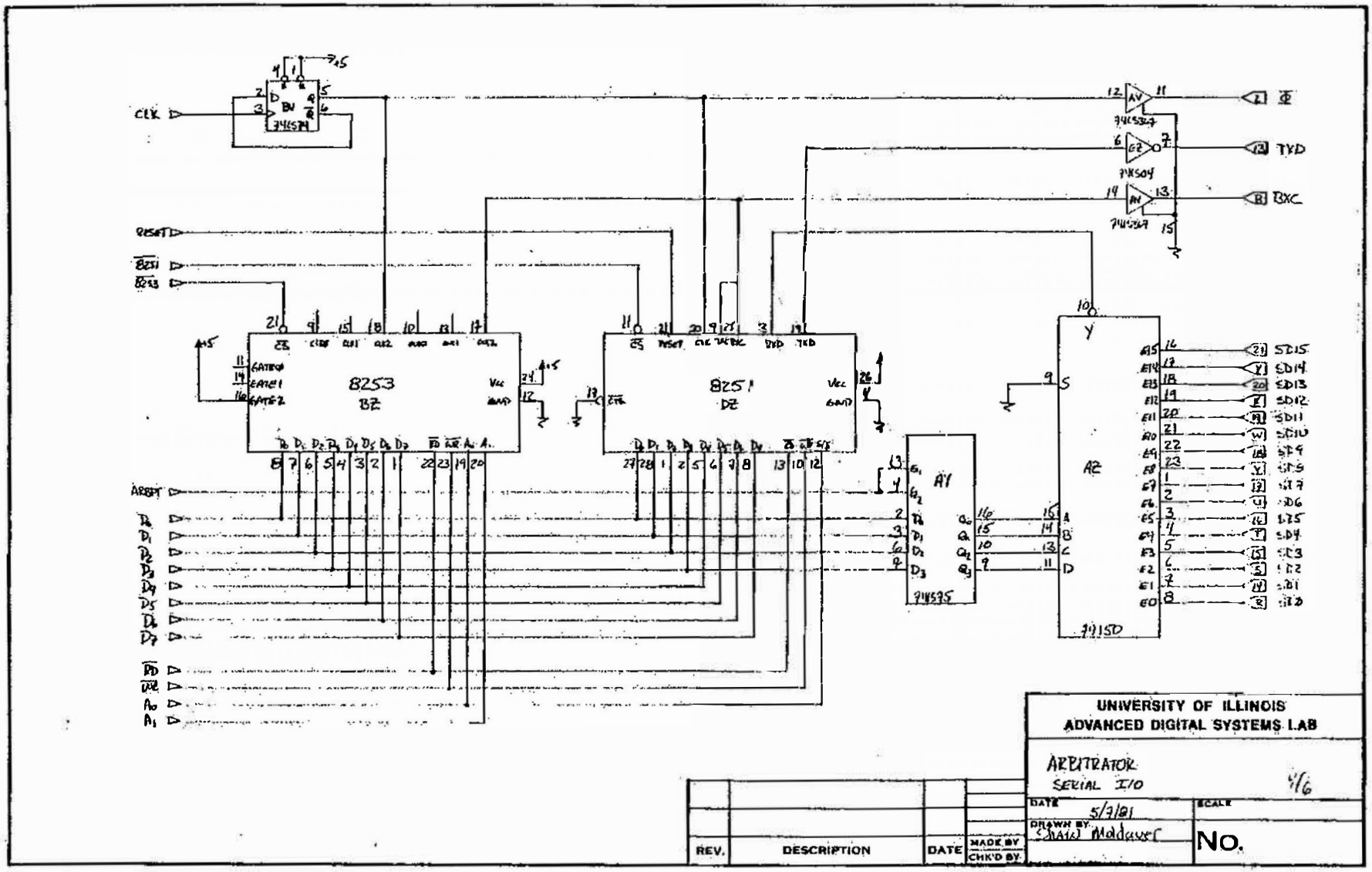


FIG. 3.3.4 ARBITRATOR SERIAL I/O

3.3.4 Input/Output Interface

The I/O capabilities of the arbitrator processor include a sixteen bit input port, a four bit output port and a four bit input port. Also the lowest four bits of the address bus and a five bit data bus are connected to each card in the multiprocessor system rack. The sixteen bit input port allows the arbitrator processor to poll the request lines from the node microcomputers. One request line comes to the port from each of the sixteen node microcomputers. These requests are not bussed on the multiprocessor backplane (see backplane). The four bit output port is a latched output that drives a four to sixteen line demultiplexer. The sixteen demultiplexer outputs are sent to the node microcomputers. These lines are used to acknowledge a requesting node microcomputer. The four bit input port is used by the arbitrator to determine in which location of the card rack the arbitrator has been placed.

The arbitrator processor can read latched data and latch new data into the node microcomputer's select latch. This is done when the 8085 CPU accesses any location from 1000H to 100FH; the low four bits of the address bus, ADR0-ADR3 select one of the node microcomputers and the read and write lines, NRD/ and NWR/, perform the desired transaction using the data bus, DAT0/-DAT4/. When a write is performed, the low four bits of the data bus are gated out

onto the system backplane. When a read is performed, the four bits of data are gated from the node microcomputer onto the low four bits of the 8085 data bus. The fifth data bit, DAT4/, is used to determine whether the addressed node microcomputer is in the rack, and is not significant when the arbitrator writes to a node microcomputer. The arbitrator I/O interface is shown in figure 3.3.5 and the bus connections and board layout are shown in figure 3.3.6.

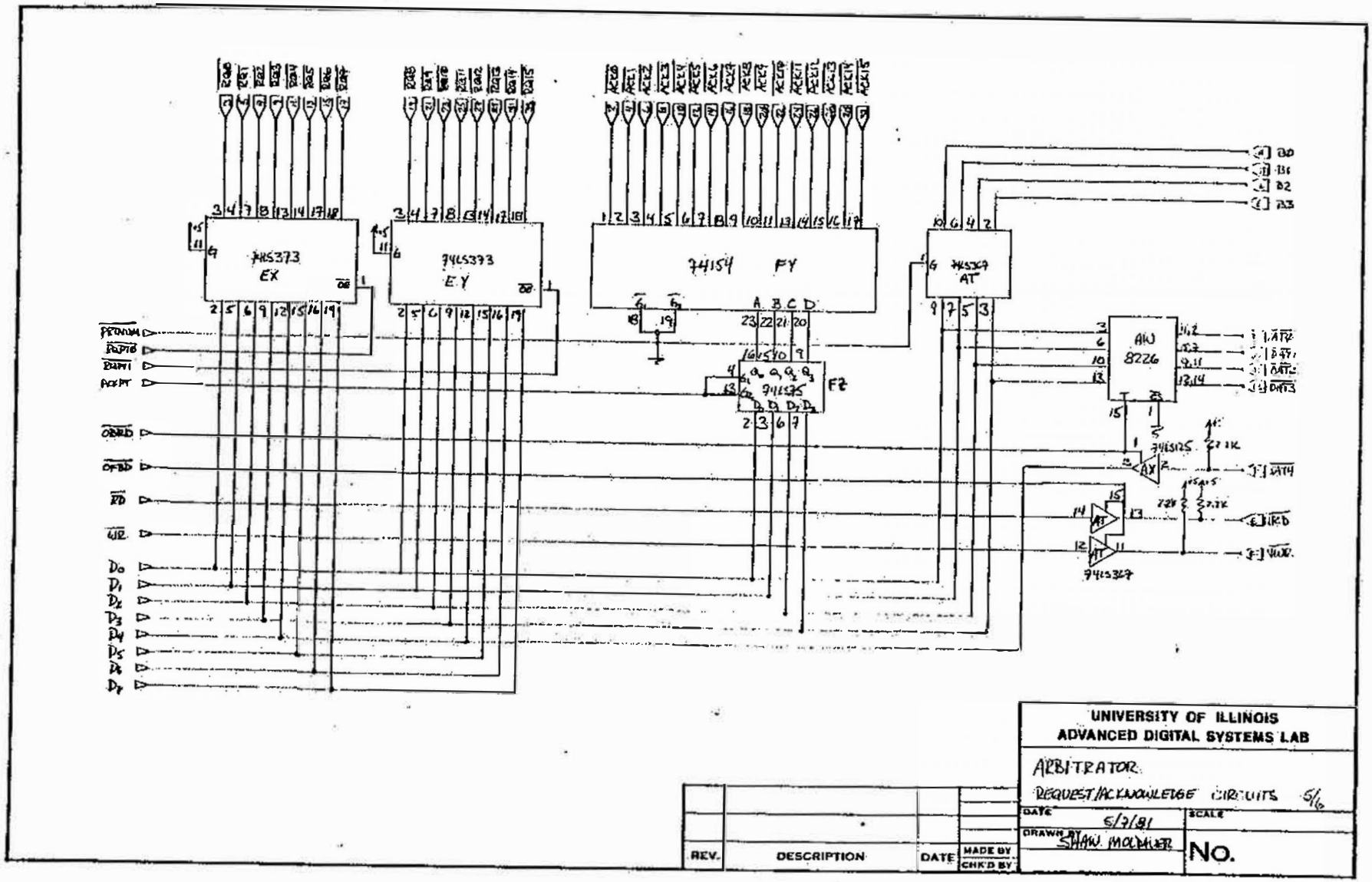


FIG. 3.3.5 ARBITRATOR I/O INTERFACE

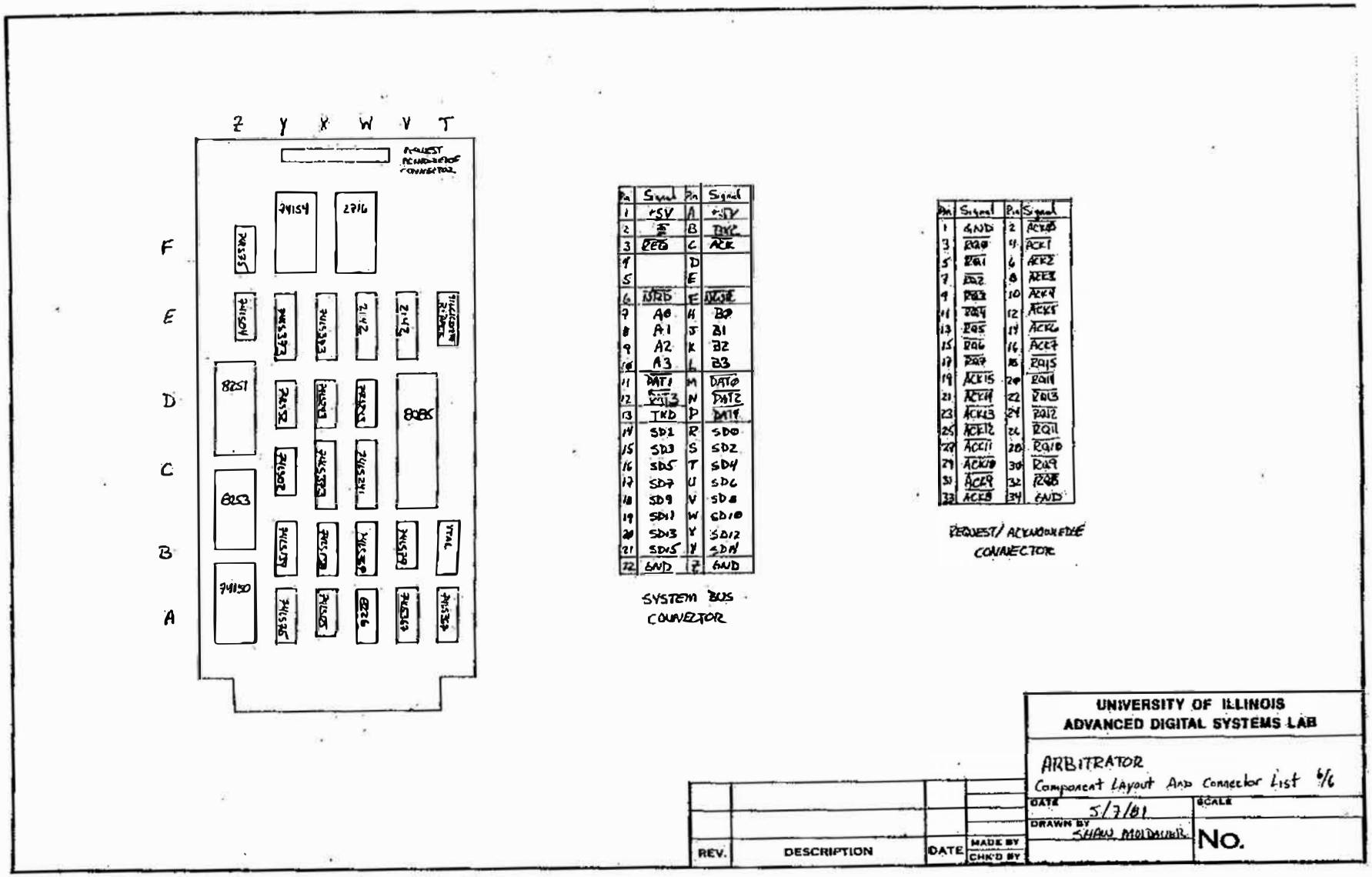


FIG. 3.3.6 ARBITRATOR BOARD LAYOUT

3.4 Backplane, Rack and Power Supply

This section completes the discussion of the microcomputer network hardware. The three types of boards plug into connectors on the backplane and the backplane then routes the signals from the boards to the appropriate destination. The rack supports and encloses the backplane and the boards used in the system. The power supply is external to the rack and supplies all the energy necessary for network operation.

3.4.1 Backplane

All of the cards that make up the multiprocessing system are interconnected through the backplane. There are two different types of backplane connections; common and individual. The signals that are common to every card in the system can be divided into three groups; data, address and serial data lines.

The arbitrator provides four control signals that are available at every card location; NRD/, NWR/, Clock and Baud clock. The data bus, DAT0/-DAT4/, is a bidirectional data path between the arbitrator processor and any selected node microcomputer. There are two address busses in the system; ADR0-ADR3 and B0-B3. The ADR bus is driven by the arbitrator processor and indicates which node

is to be accessed, the B bus, which is hard wire coded at each connector, is used by any node microcomputer to determine the location that it occupies. Sixteen serial lines are bussed on the backplane. Each of these serial lines is driven by one of the three types of cards. The serial data out of any USART is gated onto bus signal Trans Dat. This bus signal is hard wired to one of the serial lines on the backplane. A card location will have the Trans Dat signal connected to the serial line corresponding to its B address.

There are only two signals that are individual to each card location, a request line and an acknowledge line. The request and acknowledge lines are sent through a flat cable between the arbitrator processor and each card location.

Two additional lines are bussed on the backplane. These lines provide all necessary power to the cards for system operation. Since all cards are built to require only one voltage level, five volts and ground potential are bussed to each card location in the rack. The bus signal connections in the system are listed in table 3.4.1.

TABLE 3.4.1 BUS SIGNAL CONNECTIONS

Pin	Signal	Pin	Signal
1	+5 volts D.C.	A	+5 Volts D.C.
2	Clock	B	Baud Clock
3	Request/	C	Acknowledge/
4	n.c.	D	n.c.
5	n.c.	E	n.c.
6	NRD/	F	NWR/
7	ADR0	H	B0
8	ADR1	J	B1
9	ADR2	K	B2
10	ADR3	L	B3
11	DAT1/	M	DAT0/
12	DAT3/	N	DAT2/
13	Trans Dat	P	DAT4/
14	SD1	R	SD0
15	SD3	S	SD2
16	SD5	T	SD4
17	SD7	U	SD6
18	SD9	V	SD8
19	SD11	W	SD10
20	SD13	X	SD12
21	SD15	Y	SD14
22	Ground	Z	Ground

3.4.2 Rack

The rack comprises two rack sections, and each of the two are functionally equivalent. Each rack section has edge connectors and card guides for ten cards. Up to nine node microcomputers and the arbitrator can be run in one rack section, while expansion to fifteen node microcomputers requires use of both sections. The rack is built from parts of previous computer systems and is compatible with the standard nineteen inch rack mount.

3.4.3 Power Supply

Since the system was intentionally designed using only circuits that require a single five volt power supply, only one voltage level is necessary for operation. Each node microcomputer will use around one ampere of the five volt supply during stressful conditions. The arbitrator requires about one ampere. The terminal interface board uses only three hundred miliamperes. The supply used during debug and test sessions was a POWER-ONE model 5-12 five volt power supply that is capable of delivering twelve amperes of current. This supply is adequate for a system of one arbitrator, one serial interface and eight node processors. A system that contains the maximum number of cards should have a

supply that is capable of delivering around twenty amperes at a five volt level.

CHAPTER 4

SYSTEM SOFTWARE

The software for the network may be divided into three groups; resident programs that run on the node microcomputer, resident programs that run on the arbitrator, and user generated programs. The 8748 monitor is the resident node microcomputer program, it is burned into the EPROM on the 8748 microcomputer chip, and contains programs for construction and testing of user code along with network interaction subroutines. The arbitrator software controls the interconnection of the network and is burned into the 2716 EPROM on the arbitrator board. User programs are written for the 8748 node microcomputer and reside in the 2716 EPROM on the node microcomputer board.

4.1 8748 Monitor

The 8748 monitor is the basic operating software for the node microcomputer card. When a node microcomputer is reset, the 8748 microcomputer begins executing the 8748 monitor. The monitor contains initialization routines, basic input/output routines, and routines that allow network interaction. In addition to routines available to the programmer, the 8748 monitor has several useful commands which may be invoked through the serial port.

The listing for the node microcomputer monitor is in appendix A. There are four basic commands of the 8748 monitor: examine and modify, read file, write file, and execute subroutine. The examine and modify command allows the user to examine any external program memory location and optionally modify that location. The read file command allows the node microcomputer to receive and store in memory any file in Intel hex format. The write command allows any continuous region of program memory to be output in Intel hex format. The last command is the execute command which allows the user to force the execution of a program or subroutine.

During system initialization, the 8251 USART on the node processor card is programmed for full duplex serial operations and a prompt is sent out on the serial line. The monitor then waits for one of the four basic commands to be invoked. After completion of any one command the monitor then waits again for a new command.

The 8748 monitor has several basic input and output subroutines that are available to the user; these are listed in detail in appendix B. These routines include a character input command and a character output, input and hex conversion, hex conversion and output, packing and unpacking of hex numbers, and a twelve bit variable input. Also subroutines are available for gaining status information about the interconnection of the network, requesting to become interconnected with a specific device in the network, and becoming disconnected from the network.

4.2 Arbitrator Monitor

Two programs were written for the arbitrator processor. The first of these programs was a simple serial I/O based monitor that allowed hardware debug of the arbitrator processor board. The second program is the arbitrator software. The simple serial monitor is based on a 8080 single board computer monitor. It allows the user to perform basic tests from a video terminal. The instruction repertoire includes the following commands: examine and

modify memory, examine and modify processor register, execute user program, display memory, and move memory. The listing for the arbitrator software is in appendix C.

The arbitrator software is a dedicated program that allows the nodes of the network to become interconnected. The flow of the program is as follows: on power-up or reset the arbitrator initializes all internal variables and then enters the main program loop. The arbitrator will sample all the incoming request lines and then it will satisfy each request that is active before sampling the request lines again. When a request line has been found active, the arbitrator will calculate the port number of the requesting node. The serial lines of the arbitrator and the requesting nodes will be set to a circular communication path. The arbitrator will acknowledge the requesting node. The node microcomputer then submits a processor request byte serially to the arbitrator. Finally the arbitrator will determine what action has been requested. Three basic requests can be satisfied; status request, interconnection request and disconnection request. Request and acknowledge codes are listed in tables 4.2.1 and 4.2.2 respectively.

When a status request is performed, the node microcomputer sends, as a parameter, the node location that it is requesting status from. The arbitrator then determines the interconnection of the port indicated and sends back this information. When an

TABLE 4.2.1 REQUEST CODES

0XH	- No action.
2NH	- Request status of node N.
3NH	- Request connect to node N.
4XH	- Disconnect from network.

TABLE 4.2.2 ACKNOWLEDGE CODES

0XH	- No action.
2NH	- Status acknowledge, node is connected to node N.
3NH	- Connect to node N.
4XH	- Disconnect acknowledge.
5NH	- No connect, node N busy.
6NH	- Status acknowledge, node N not in system.
7NH	- No connect, node N not in system.

note: all numbers given in hex (hence H).

X means don't care.

N is hex for any node 0-F.

interconnection request is made, the node microcomputer sends as a parameter the node to which it requests connection. The requested node could be either in use, not in the rack or available for communications. An appropriate response is then sent back to the requesting node, and if the node was available, the node will be connected when the acknowledge signal becomes inactive. If the requested node is not available, the arbitrator waits for another request, either a new connection, a status check or a disconnection. When a node microcomputer requests to become disconnected from the network, the arbitrator disconnects the requesting node and then deactivates the corresponding acknowledge line.

4.3 Programming

The user must configure the system by designing software for the node microcomputer. The size constraints on the software are those of the memory address space. A socket for 2048 bytes of EPROM is available for program storage. The address for this EPROM is 800H through FFFH in the 8748 node microcomputers. During program debug, the user may prefer RAM in this 2048 byte region. This is easily done with the switch on the node microcomputer front panel. System subroutines in the 8748 monitor on the 8748 chip are available to the user and are listed in appendix B. The 8748 monitor does not use interrupts, but it does allow the user to specify interrupt service routines. Upon detection of an interrupt, the 8748 microcomputer saves the execution address and jumps to either of two locations, depending on whether the interrupt is external or generated by the internal timer. In either case, the 8748 monitor will switch to the alternate register bank and save the accumulator in alternate register 7. If the interrupt is external, program execution will continue at location 7F0H in external RAM. If the interrupt is from the internal timer, execution will continue at location 7F4H in external RAM. The user must place either a jump to a service routine or the actual service routine at these locations in external RAM. Upon completion of the service routine, the accumulator must be restored and the normal register bank selected.

Locations 7F0H through 7F7H may be used for variable storage only if interrupts are not used, but it is suggested that these locations not be used for anything except interrupt vectors. The monitor does use additional memory locations for storage, the top two bytes of data storage internal to the 8748 microcomputer are used during the hex file read command and will be changed each time this command is invoked. Also locations 7F8H through 7FFH in external program memory are reserved for monitor use and may not be changed by the system user. The user may then allocate program memory and variable storage from locations 400H through 7EFH and from locations 800H through FFFH in external program memory. Variable storage is also available at locations 20H through 3DH in the internal data memory.

CHAPTER 5

CONCLUDING REMARKS

The microcomputer network has been constructed and tested. The possible applications of this project are many and varied. The individual boards were all tested before the network was assembled, and were found totally functional. The existing system software has also been tested in the individual boards. When the integration of the network was complete, the node microcomputers were loaded with test routines to check the reconfiguration capabilities of the network. These routines were also found functional and therefore proved the network operational.

A system consisting of one arbitrator processor, one terminal interface board, and seven node microcomputers now is running and available for application. A possible application for the microcomputer network would be an environment with a restriction on physical size, yet requiring control of a number of

output devices and sensing of a number of input devices. The microcomputer network would be a good choice for controlling a totally self-contained robot system. Each node microcomputer could be assigned the control of a different aspect of the system. One could control the motion of the system while another might control one or two types of position sensors. With each node microcomputer assigned a different task, the actions of the system would be a result of the interactions of the nodes in the microcomputer network.

Additional functions could easily be added to the microcomputer network. The network will satisfy the requirements of the originally intended application, and addition of nodes interrupting each other, system self startup or other features were not included because they might interfere with certain applications.

APPENDIX A

8748 MONITOR LISTING

```
LINE  ADDR  B1 B2 B3 B4 ERROR          8048 MACRO ASSEMBLER VER 2.0                PAGE    1

   2                                                    ;8748 MONITOR FOR NODE MICROCOMPUTER
   3                                    ;
   4                                    ;
   5                                    ;
   6  003A                              PRMPT   EQU     ':'         ;PROMPT CHARACTER
   7  07FA                              CLINST  EQU     07FAH
   8  00F1                              USAB    EQU     0F1H        ;UART STATUS AND MASK
   9  0001                              USOB    EQU     01H         ;UART STATUS OR MASK
  10  00F0                              UDAB    EQU     0F0H        ;UART DATA AND MASK
  11  0002                              RDRDY   EQU     02H         ;RECEIVER READY MASK
  12  0004                              TXE     EQU     4           ;TRANSMITTER EMPTY MASK
  13  0001                              TRRDY   EQU     01H         ;TRANSMITTER READY MASK
  14  003E                              OFST    EQU     3EH         ;LOCATION WHERE OFFSET FOR READ IS STORE
  15  000D                              CR      EQU     0DH         ;ASCII CARRIAGE RETURN
  16  0020                              SP      EQU     20H         ;ASCII SPACE
  17  000A                              LF      EQU     0AH         ;ASCII LINE FEED
  18  003F                              ERCR    EQU     '?'         ;ERROR CHARACTER
  19  07F0                              INTRPT  EQU     07F0H       ;ADDRESS OF INTERRUPT SERVICE ROUTINE
  20  07F4                              CTINT   EQU     07F4H       ;ADDRESS OF TIMER INTERRUPT ROUTINE
  21  0025                              CMD     EQU     025H        ;UART COMMAND INSTRUCTION
  22  00CE                              MODE    EQU     0CEH        ;UART MODE INSTRUCTION
  23  00C0                              SRST    EQU     0C0H        ;START UART RESET
  24  0041                              ERST    EQU     041H        ;END UART RESET
  25                                    ;
  26                                    ;
  27                                    ;
```

```
 29                                    ORG     0000H
 30  0000  44 35                       JMP     INIT
 31  0002  00                          NOP
 32  0003  D5                          SEL     RB1             ;SWITCH TO ALT REGESTERS AND SAVE A
 33  0004  AF                          MOV     R7,A
 34  0005  E4 F0                       JMP     INTRPT          ;SERVICE INTERUPT
 35  0007  D5                          SEL     RB1             ;SWITCH TO ALT REGS AND SAVE A
 36  0008  AF                          MOV     R7,A
 37  0009  E4 F4                       JMP     CTINT           ;SERVICE TIMER INTERRUPT
 38                                                            ;JUMP TABLE FOR ROUTINE ENTRY
 39  000B  04 2B                 JTBL: JMP     CIN             ;CHARACTER INPUT ROUTINE
 40  000D  04 3A                       JMP     COUT            ;CHARACTER OUTPUT ROUTINE
 41  000F  04 94                       JMP     CRLF            ;SEND CARRIAGE RETURN-LINE FEED
 42  0011  04 65                       JMP     OUTHEX          ;INPUT HEX CHARACTER
 43  0013  04 76                       JMP     INBYTE          ;INPUT TWO HEX CHARACTERS
 44  0015  04 86                       JMP     OUTBYTE         ;OUTPUT TWO HEX CHARACTERS
 45  0017  04 9D                       JMP     INAD            ;GET ADDRESS OR DATA SPECIFICATION
 46  0019  04 BD                       JMP     XREAD           ;READ EXTERNAL RAM
 47  001B  04 C8                       JMP     XWRITE          ;WRITE TO EXTERNAL RAM
 48  001D  44 2A                       JMP     ERROR           ;ENTER MONITOR WITH ERROR
 49  001F  44 54                       JMP     PROMPT          ;ENTER MONITOR WITH NO ERROR
 50  0021  44 84                       JMP     CTA             ;CONNECT TO ARBITRATOR
 51  0023  44 AC                       JMP     DCNCT           ;DISCONNECT FROM NETWORK
 52  0025  44 B7                       JMP     CNNCT           ;CONNECT TO SPECIFIED NODE
 53  0027  24 A5                       JMP     WR10            ;WRITE SUBROUTINE
 54  0029  44 C4                       JMP     WFCNCT          ;CONNECT TO SPECIFIED NODE BUT
 55                                                            ;WAIT TILL NODE AVAILABLE
```

```
  57                            ;
  58                            ;
  59                            ;
  60                            ; * CIN - GETS CHARACTER FROM USART TO A REGISTER
  61                            ;         DESTROYS - OUTPUT FROM P2
  62                            ;
  63                            ;
  64  0028  9A F1               CIN:    ANL     P2,#USAB
  65  002D  8A 01                       ORL     P2,#USOB        ;ADDRESS STATUS OF USART PORT
  66  002F  80                  CIN05:  MOVX    A,@R0           ;READ USART STATUS
  67  0030  53 02                       ANL     A,#RDRDY        ;IS CHARACTER IN ?
  68  0032  C6 2F                       JZ      CIN05           ;LOOP TILL CHARACTER
  69  0034  9A F0                       ANL     P2,#UDAB        ;ADDRESS DATA OF USART PORT
  70  0036  80                          MOVX    A,@R0           ;PUT DATA IN ACCUMILATOR
  71  0037  53 7F                       ANL     A,#7FH          ;MASK PARITY
  72  0039  83                          RET
  73                            ;
  74                            ;
  75                            ; * COUT - OUTPUTS CHARACTER IN A REGISTER TO USART
  76                            ;          DESTROYS - R2, OUTPUT FROM P2
  77                            ;
  78                            ;
  79  003A  9A F1               COUT:   ANL     P2,#USAB
  80  003C  8A 01                       ORL     P2,#USOB        ;ADDRESS STATUS OF USART PORT
  81  003E  AA                          MOV     R2,A            ;SAVE CHARACTER IN R2
  82  003F  80                  COUT05: MOVX    A,@R0           ;READ USART STATUS
  83  0040  53 01                       ANL     A,#TRRDY        ;IS TRANSMITTER READY ?
  84  0042  C6 3F                       JZ      COUT05
  85  0044  9A F0                       ANL     P2,#UDAB        ;POINT TO DATA
  86  0046  FA                          MOV     A,R2            ;GET CHARACTER TO A REGISTER
  87  0047  90                          MOVX    @R0,A           ;OUTPUT CHARACTER
  88  0048  83                          RET
```

```
  90                            ;
  91                            ;
  92                            ; * INHEX - GETS CHARACTER FROM USART AND CONVERTS TO HEX
  93                            ;           RETURNS HEX IN A REGISTER
  94                            ;           RETURNS WITH F0 SET IF NOT VALID HEX CHARACTER
  95                            ;           DESTROYS - R2,OUTPUT FROM P2
  96                            ;
  97                            ;
  98  0049  85                 INHEX:  CLR     F0
  99  004A  14 28                       CALL    CIN
 100  004C  14 3A                       CALL    COUT
 101  004E  03 C6                       ADD     A,#0C6H
 102  0050  F6 57                       JC      INH05
 103  0052  03 0A                       ADD     A,#0AH
 104  0054  E6 63                       JNC     INH10
 105  0056  83                         RET
 106  0057  FA                 INH05:  MOV     A,R2
 107  0058  03 B9                       ADD     A,#0B9H
 108  005A  F6 63                       JC      INH10
 109  005C  03 06                       ADD     A,#06H
 110  005E  E6 63                       JNC     INH10
 111  0060  03 0A                       ADD     A,#0AH
 112  0062  83                         RET
 113  0063  95                 INH10:  CPL     F0
 114  0064  83                         RET
 115                            ;
 116                            ;
 117                            ; * OUTHEX - OUTPUTS THE LOWER FOUR BITS OF THE A REGISTER AS HEX CHAR.
 118                            ;           DESTROYS - R2,OUTPUT FROM P2
 119                            ;
 120                            ;
 121  0065  53 0F               OUTHEX: ANL     A,#0FH          ;MASK OFF UPPER FOUR BITS
 122  0067  AA                         MOV     R2,A            ;SAVE IN R2
 123  0068  03 F6                       ADD     A,#0F6H         ;SUBTRACT 10
 124  006A  F6 71                       JC      OH05            ;IF NOT LESS THAN 9 JUMP
 125  006C  FA                         MOV     A,R2            ;GET HEX
 126  006D  03 30                       ADD     A,#30H          ;ADD 30H TO MAKE ASCII NUMBER
 127  006F  04 3A                       JMP     COUT            ;JUMP TO COUT, RETURN FROM THERE
 128  0071  FA                 OH05:   MOV     A,R2            ;GET HEX
 129  0072  03 37                       ADD     A,#037H         ;ADJUST FOR ASCII LETTER
 130  0074  04 3A                       JMP     COUT            ;JUMP TO COUT, RETURN FROM THERE
```

```
                                ;
                                ;
                                ; * INBYTE - GETS TWO ASCII CHARS FROM PORT AND PACKS INTO A REGISTER
                                ;           DESTROYS - R2,R3,R7
                                ;           OUTPUTS UPDATED CHECKSUM IN R7
                                ;
                                ;
0076 14 49                      INBYTE: CALL    INHEX           ;GET A HEX CHAR
0078 B6 85                              JF0     IB05            ;IF INVALID, EXIT
007A 47                                 SWAP    A
007B AB                                 MOV     R3,A            ;STORE IN UPPER OF R3
007C 14 49                              CALL    INHEX
007E B6 85                              JF0     IB05            ;GET SECOND HEX
0080 4B                                 ORL     A,R3            ;OR TOGETHER THE TWO HEX CHARS.
0081 AB                                 MOV     R3,A            ;STORE IN R3
0082 2F                                 XCH     A,R7
0083 6F                                 ADD     A,R7
0084 2F                                 XCH     A,R7            ;UPDATE CHECKSUM
0085 83                         IB05:   RET
                                ;
                                ;
                                ; * OUTBYTE - OUTPUTS BYTE IN A REGISTER AS TWO ASCII CHARS.
                                ;           DESTROYS - R2,R3,R7
                                ;           OUTPUTS UPDATED CHECKSUM IN R7
                                ;
                                ;
0086 2F                         OUTBYTE:XCH     A,R7
0087 6F                                 ADD     A,R7
0088 2F                                 XCH     A,R7            ;UPDATE CHECKSUM
0089 AB                                 MOV     R3,A            ;SAVE IN R3
008A 53 F0                              ANL     A,#0F0H         ;MASK OFF FIRST HEX
008C 47                                 SWAP    A
008D 14 65                              CALL    OUTHEX          ;PRINT FIRST HEX
008F FB                                 MOV     A,R3            ;GET BYTE
0090 53 0F                              ANL     A,#0FH          ;MASK OFF SECOND HEX
0092 04 65                              JMP     OUTHEX          ;JUMP TO OUTHEX, RETURN FROM THERE
```

```
169                                     ;
170                                     ;
171                                     ; * CRLF - PRINTS CARRIAGE RETURN LINE FEED ON CONSOLE
172                                     ;
173                                     ;
174  0094  23 0D                      *CRLF:   MOV      A,#CR
175  0096  14 3A                              CALL     COUT
176  0098  23 0A                              MOV      A,#LF
177  009A  14 3A                              CALL     COUT
178  009C  83                                 RET
179                                     ;
180                                     ;
181                                     ; * INAD - GETS THREE HEX DIGITS FROM INPUT STREAM AND RETURNS
182                                     ;          WITH TWELVE BIT ADDRESS/DATA IN R3 AND R4
183                                     ;          F0 IS SET IF FIRST CHARACTER IS TERMINATOR
184                                     ;          TERMINATOR IS RETURNED IN R2
185                                     ;          DESTROYS - R2,R3,R4
186                                     ;
187                                     ;
188  009D  27                          INAD:    CLR      A
189  009E  AB                                   MOV      R3,A
190  009F  AC                                   MOV      R4,A
191  00A0  14 AC                       AD05:    CALL     FILL            ;GET NEXT CHAR AND SHIFT INTO R3:R4
192  00A2  B6 AB                                JF0      AD20            ;EXIT IF TERMINATOR IS FIRST CHAR
193  00A4  14 AC                       AD10:    CALL     FILL            ;GET SUBSEQUENT CHARS
194  00A6  B6 AA                                JF0      AD15            ;EXIT IF TERMINATOR
195  00A8  04 A4                                JMP      AD10            ;LOOP TILL TERMINATOR
196  00AA  85                          AD15:    CLR      F0              ;EXIT NORMALLY
197  00AB  83                          AD20:    RET                      ;EXIT IF FIRST CHAR TERMINATOR
198                                     ;
199                                     ;
200                                     ; * FILL - GETS ONE HEX NUMBER AND SHIFTS IT INTO THE 12 BIT NUMBER IN R
201                                     ;          DESTROYS - R2,R3,R4,R7
202                                     ;
203                                     ;
204  00AC  14 49                       FILL:    CALL     INHEX           ;GET NEXT CHAR
205  00AE  B6 BC                                JF0      FI05            ;RETURN IF TERMINATOR
206  00B0  AF                                   MOV      R7,A            ;STORE IN R7
207  00B1  FC                                   MOV      A,R4            ;GET LOW EIGHT BITS
208  00B2  47                                   SWAP     A
209  00B3  53 0F                                ANL      A,#0FH          ;PUT HIGH NIBBLE INTO R3
210  00B5  AB                                   MOV      R3,A
211  00B6  FC                                   MOV      A,R4            ;GET LOW EIGHT
212  00B7  47                                   SWAP     A
213  00B8  53 F0                                ANL      A,#0F0H
214  00BA  4F                                   ORL      A,R7
215  00BB  AC                                   MOV      R4,A
216  00BC  83                          FI05:    RET
```

```
                        ;
                        ;
                        ; * XREAD - READS EXTERNAL MEMORY
                        ;          LOW(ADDRESS) IN R0
                        ;          HIGH(ADDRESS) IN R6
                        ;          CONTENTS READ INTO A REGISTER
                        ;
                        ;
00BD FE                 XREAD:  MOV     A,R6
00BE 53 0F                      ANL     A,#0FH          ;GET HIGH(ADDRESS)
00C0 AE                         MOV     R6,A
00C1 9A F0                      ANL     P2,#0F0H
00C3 0A                         IN      A,P2
00C4 4E                         ORL     A,R6
00C5 3A                         OUTL    P2,A            ;PUT HIGH(ADDRESS) ON ADDRESS BUS
00C6 80                         MOVX    A,@R0           ;READ DATA
00C7 83                         RET
                        ;
                        ;
                        ; * XWRITE - WRITES DATA IN A REGISTER TO EXTERNAL MEMORY
                        ;           LOW(ADDRESS) IN R0
                        ;           HIGH(ADDRESS) IN R6
                        ;           DESTROYS - R2
                        ;
00C8 2E                 XWRITE: XCH     A,R6            ;SAVE DATA IN R3
00C9 53 0F                      ANL     A,#0FH
00CB 9A F0                      ANL     P2,#0F0H
00CD AB                         MOV     R3,A
00CE 0A                         IN      A,P2
00CF 4B                         ORL     A,R3
00D0 3A                         OUTL    P2,A            ;HIGH(ADDRESS) TO P2
00D1 2E                         XCH     A,R6            ;GET DATA BACK
00D2 90                         MOVX    @R0,A           ;WRITE DATA
00D3 83                         RET
                        ;
                        ;
                        ;
```

```
 256  00D4  44 2A              SC25:   JMP     ERROR
 257
 258
 259                           ; * SCMD
 260
 261
 262  00D6  14 9D              SCMD:   CALL    INAD            ;GET ADDRESS
 263  00D8  B6 D4                      JF0     SC25            ;ERROR EXIT
 264  00DA  FA                         MOV     A,R2            ;GET TERMINATOR
 265  00DB  D3 0D                      XRL     A,#CR
 266  00DD  96 D4                      JNZ     SC25            ;ERROR IF NOT CR
 267  00DF  FB                         MOV     A,R3            ;GET MSB OF ADDRESS
 268  00E0  AE                         MOV     R6,A            ;PUT IN R6
 269  00E1  FC                         MOV     A,R4            ;GET LSB OF ADDRESS
 270  00E2  A8                         MOV     R0,A            ;PUT IN R0
 271  00E3  14 94              SC05:   CALL    CRLF
 272  00E5  FE                         MOV     A,R6
 273  00E6  14 65                      CALL    OUTHEX          ;PRINT MSB OF ADDRESS
 274  00E8  F8                         MOV     A,R0
 275  00E9  14 86                      CALL    OUTBYTE         ;PRINT LSB OF ADDRESS
 276  00EB  23 20                      MOV     A,#SP
 277  00ED  14 3A                      CALL    COUT
 278  00EF  14 3A                      CALL    COUT            ;PRINT TWO SPACES
 279  00F1  23 20              SC10:   MOV     A,#SP
 280  00F3  14 3A                      CALL    COUT            ;PRINT A SPACE
 281  00F5  14 BD                      CALL    XREAD           ;GET CONTENTS OF MEMORY
 282  00F7  14 86                      CALL    OUTBYTE         ;PRINT CONTENTS
 283  00F9  23 2D                      MOV     A,#'-'
 284  00FB  14 3A                      CALL    COUT
 285  00FD  14 9D                      CALL    INAD            ;GET SUBSTITUTION
 286  00FF  B6 04                      JF0     SC15            ;JUMP IF NO SUBSTITUTION
 287  0101  FC                         MOV     A,R4            ;GET LSB OF DATA TO A REGISTER
 288  0102  14 C8                      CALL    XWRITE          ;REPLACE MEMORY
 289  0104  FA                 SC15:   MOV     A,R2
 290  0105  D3 0D                      XRL     A,#CR           ;WAS TERMINATOR CR?
 291  0107  C6 1F                      JZ      SC20            ;RETURN IF CR
 292  0109  FA                         MOV     A,R2
 293  010A  D3 20                      XRL     A,#SP
 294  010C  96 67                      JNZ     RC20            ;ERROR IF NOT SP OR CR
 295  010E  F8                         MOV     A,R0
 296  010F  03 01                      ADD     A,#01H          ;INC ADDRESS
 297  0111  A8                         MOV     R0,A
 298  0112  FE                         MOV     A,R6
 299  0113  13 00                      ADDC    A,#00H
 300  0115  AE                         MOV     R6,A
 301  0116  F8                         MOV     A,R0
 302  0117  53 07                      ANL     A,#07H          ;WHEN ADDRESS IS 8*N, THEN NEW LINE
 303  0119  96 1D                      JNZ     SC30
 304  011B  04 E3                      JMP     SC05
 305  011D  04 F1              SC30:   JMP     SC10
 306  011F  83                 SC20:   RET
```

```
 308                              ; * RDCMD - READS INTEL HEX FILE
 309  0120  14 9D                 RDCMD:  CALL    INAD            ;GET OFFSET
 310  0122  FA                             MOV     A,R2
 311  0123  D3 3A                         XRL     A,#':'
 312  0125  C6 3B                         JZ      RC10
 313  0127  FA                             MOV     A,R2
 314  0128  D3 0D                         XRL     A,#CR
 315  012A  96 67                         JNZ     RC20
 316  012C  B8 3E                         MOV     R0,#OFST
 317  012E  FB                             MOV     A,R3
 318  012F  A0                             MOV     @R0,A           ;STORE OFFSET
 319  0130  18                             INC     R0
 320  0131  FC                             MOV     A,R4
 321  0132  A0                             MOV     @R0,A
 322  0133  14 2B                 RC05:   CALL    CIN
 323  0135  14 3A                         CALL    COUT
 324  0137  D3 3A                         XRL     A,#':'
 325  0139  96 33                         JNZ     RC05            ;LOOP TILL RECORD MARK
 326  013B  27                     RC10:   CLR     A
 327  013C  AF                             MOV     R7,A            ;CLEAR CHECKSUM
 328  013D  14 76                         CALL    INBYTE          ;GET LENGTH
 329  013F  AD                             MOV     R5,A            ;STORE LENGTH IN R5
 330  0140  14 76                         CALL    INBYTE          ;GET MSB OF ADDRESS
 331  0142  AE                             MOV     R6,A            ;STORE IN R6
 332  0143  14 76                         CALL    INBYTE          ;GET LSB OF ADDRESS
 333  0145  A8                             MOV     R0,A            ;STORE IN R0
 334  0146  14 76                         CALL    INBYTE          ;GET RECORD TYPE
 335  0148  96 59                         JNZ     RC25            ;EXIT IF END OF FILE
 336  014A  B9 3F                         MOV     R1,#OFST+1
 337  014C  F1                             MOV     A,@R1           ;GET LSB OF OFFSET
 338  014D  68                             ADD     A,R0
 339  014E  A8                             MOV     R0,A            ;ADD OFFSET TO ADDRESS
 340  014F  C9                             DEC     R1
 341  0150  F1                             MOV     A,@R1           ;GET MSB OF OFFSET
 342  0151  7E                             ADDC    A,R6            ;ADD OFFSET TO ADDRESS
 343  0152  AE                             MOV     R6,A
 344  0153  14 76                 RC15:   CALL    INBYTE          ;READ DATA FROM PORT
 345  0155  14 C8                         CALL    XWRITE          ;WRITE DATA INTO MEMORY
 346  0157  F8                             MOV     A,R0            ;INCREMENT ADDRESS
 347  0158  03 01                         ADD     A,#01H
 348  015A  A8                             MOV     R0,A
 349  015B  FE                             MOV     A,R6
 350  015C  13 00                         ADDC    A,#00H
 351  015E  AE                             MOV     R6,A            ;REPLACE  ADDRESS
 352  015F  ED 53                         DJNZ    R5,RC15         ;LOOP IF CHARACTERS LEFT
 353  0161  14 76                         CALL    INBYTE          ;GET CHECKSUM
 354  0163  27                             CLR     A
 355  0164  6F                             ADD     A,R7
 356  0165  C6 33                         JZ      RC05
 357  0167  44 2A                 RC20:   JMP     ERROR           ;EXIT IF BAD CHECKSUM
 358  0169  14 76                 RC25:   CALL    INBYTE          ;READ CHECKSUM FOR END OF FILE RECORD
 359  016B  83                             RET
```

```
                                ;
                                ;
                                ; * WRITE - WRITES MEMORY TO USART IN INTEL HEX FORMAT
                                ;
                                ;
016C 14 9D          WRCMD:  CALL    INAD            ;GET STARTING ADDRESS
016E B6 F8                  JFO     WR35
0170 FB                     MOV     A,R3            ;PUT STARTING ADDRESS INTO R0:R6
0171 AE                     MOV     R6,A
0172 FC                     MOV     A,R4
0173 A8                     MOV     R0,A
0174 FA                     MOV     A,R2            ;GET TERMINATOR
0175 D3 20                  XRL     A,#SP           ;IF SPACE THEN END ADDRESS
0177 C6 8D                  JZ      WR05
0179 FA                     MOV     A,R2
017A D3 53                  XRL     A,#'S'          ;IF 'S' THEN SWATH
017C 96 F8                  JNZ     WR35            ;IF NOT SP OR 'S' THEN ERROR
017E 14 9D                  CALL    INAD            ;GET SWATH
0180 B6 F8                  JFO     WR35
0182 FA                     MOV     A,R2
0183 D3 0D                  XRL     A,#CR
0185 96 F8                  JNZ     WR35            ;IN TERMINATOR NOT CR, THEN ERROR
0187 FC                     MOV     A,R4            ;PUT R3:R4 INTO R4:R5
0188 AD                     MOV     R5,A
0189 FB                     MOV     A,R3
018A AC                     MOV     R4,A
018B 24 A5                  JMP     WR10
018D 14 9D          WR05:   CALL    INAD            ;GET END ADDRESS
018F B6 F8                  JFO     WR35
0191 FA                     MOV     A,R2
0192 D3 0D                  XRL     A,#CR
0194 96 F8                  JNZ     WR35            ;ERROR IF TERMINATOR NOT CR
0196 FE                     MOV     A,R6            ;SUBTRACT START ADDRESS FROM END ADDRESS
0197 37                     CPL     A               ;AND PUT RESULT IN SWATH REGISTERS R4:R5
0198 17                     INC     A
0199 A9                     MOV     R1,A
019A F8                     MOV     A,R0
019B 37                     CPL     A
019C 17                     INC     A
019D 17                     INC     A
019E 6C                     ADD     A,R4
019F AD                     MOV     R5,A
01A0 F9                     MOV     A,R1
01A1 7B                     ADDC    A,R3
01A2 AC                     MOV     R4,A
01A3 E6 F8                  JNC     WR35
01A5 14 94          WR10:   CALL    CRLF            ;POSITION TO NEW LINE
01A7 23 3A                  MOV     A,#':'
01A9 14 3A                  CALL    COUT            ;PRINT RECORD MARK
01AB 27                     CLR     A
01AC AF                     MOV     R7,A            ;CLEAR CHECKSUM
01AD FD                     MOV     A,R5            ;GET LSB(SWATH)
```

```
 414  01AE  03 F0              ADD     A,#0F0H         ;SUBTRACT SIXTEEN
 415  01B0  AD                 MOV     R5,A            ;REPLACE SWATH
 416  01B1  FC                 MOV     A,R4            ;GET MSB(SWATH)
 417  01B2  13 FF              ADDC    A,#0FFH
 418  01B4  AC                 MOV     R4,A            ;REPLACE SWATH
 419  01B5  E6 BB              JNC     WR15            ;IF LESS THAN SIXTEEN LEFT, JUMP
 420  01B7  23 10              MOV     A,#10H          ;SET A REGISTER TO 11
 421  01B9  24 C0              JMP     WR20
 422  01BB  27                 WR15:   CLR     A
 423  01BC  AC                 MOV     R4,A            ;CLEAR MSB(SWATH)
 424  01BD  2D                 XCH     A,R5            ;GET REMAINDER AND CLEAR LSB(SWATH)
 425  01BE  03 10              ADD     A,#10H          ;ADD 16 TO REMAINDER
 426  01C0  C6 E5              WR20:   JZ      WR30            ;IF NONE LEFT, WRITE END OF FILE
 427  01C2  A9                 MOV     R1,A            ;STORE COUNT IN R1
 428  01C3  14 86              CALL    OUTBYTE         ;PRINT BYTE COUNT
 429  01C5  FE                 MOV     A,R6            ;GET MSB(ADDRESS)
 430  01C6  14 86              CALL    OUTBYTE         ;PRINT MSB(ADDRESS)
 431  01C8  F8                 MOV     A,R0
 432  01C9  14 86              CALL    OUTBYTE         ;PRINT LSB(ADDRESS)
 433  01CB  27                 CLR     A
 434  01CC  14 86              CALL    OUTBYTE         ;PRINT RECORD TYPE
 435  01CE  14 BD              WR25:   CALL    XREAD           ;READ CONTENTS OF MEMORY
 436  01D0  14 86              CALL    OUTBYTE         ;PRINT DATA
 437  01D2  28                 XCH     A,R0            ;POINT TO NEXT ADDRESS
 438  01D3  03 01              ADD     A,#01H
 439  01D5  28                 XCH     A,R0
 440  01D6  2E                 XCH     A,R6
 441  01D7  13 00              ADDC    A,#00H
 442  01D9  53 0F              ANL     A,#0FH
 443  01DB  2E                 XCH     A,R6
 444  01DC  E9 CE              DJNZ    R1,WR25         ;LOOP TILL LINE OUTPUTED
 445  01DE  FF                 MOV     A,R7            ;GET CHECKSUM
 446  01DF  37                 CPL     A
 447  01E0  17                 INC     A
 448  01E1  14 86              CALL    OUTBYTE         ;PRINT CHECKSUM
 449  01E3  24 A5              JMP     WR10            ;NEXT LINE
 450  01E5  27                 WR30:   CLR     A
 451  01E6  14 86              CALL    OUTBYTE         ;PRINT RECORD LENGTH
 452  01E8  27                 CLR     A
 453  01E9  14 86              CALL    OUTBYTE         ;PRINT ADDRESS
 454  01EB  27                 CLR     A
 455  01EC  14 86              CALL    OUTBYTE
 456  01EE  23 01              MOV     A,#01H
 457  01F0  14 86              CALL    OUTBYTE         ;PRINT RECORD TYPE
 458  01F2  FF                 MOV     A,R7
 459  01F3  37                 CPL     A
 460  01F4  17                 INC     A
 461  01F5  14 86              CALL    OUTBYTE
 462  01F7  83                 RET
 463  01F8  44 2A              WR35:   JMP     ERROR
```

```
                                ;
                                ;
                                ; * EXECUTE
                                ;
                                ;
01FA 14 9D              EXEC:   CALL    INAD            ;GET ADDRESS
01FC B6 F8                      JF0     WR35            ;ERROR IF WRONG FORMAT
01FE FA                         MOV     A,R2
01FF D3 0D                      XRL     A,#CR           ;TERMINATOR=CR?
0201 96 2A                      JNZ     ERROR
0203 0A                         IN      A,P2
0204 53 F0                      ANL     A,#0F0H
0206 43 07                      ORL     A,#.HIGH.CLINST
0208 3A                         OUTL    P2,A
0209 FB                         MOV     A,R3
020A 72 10                      JB3     EX05
020C 23 E5                      MOV     A,#0E5H         ;SEL MB0 COMMAND
020E 44 12                      JMP     EX10
0210 23 F5              EX05:   MOV     A,#0F5H         ;SEL MB1 COMMAND
0212 B8 FA              EX10:   MOV     R0,#.LOW.CLINST
0214 90                         MOVX    @R0,A           ;STORE SEL MB COMMAND
0215 FB                         MOV     A,R3
0216 47                         SWAP    A
0217 97                         CLR     C
0218 F7                         RLC     A
0219 43 14                      ORL     A,#14H          ;CREATE CALL INSTRUCTION
021B 18                         INC     R0              ;STORE AT LOCATION CLINST+1
021C 90                         MOVX    @R0,A
021D 18                         INC     R0              ;POINT TO NEXT LOCATION
021E FC                         MOV     A,R4
021F 90                         MOVX    @R0,A           ;STORE PAGE ADDRESS
0220 18                         INC     R0
0221 23 44                      MOV     A,#44H
0223 90                         MOVX    @R0,A           ;STORE RETURN INSTRUCTION
0224 18                         INC     R0
0225 23 30                      MOV     A,#.LOW.ER05
0227 90                         MOVX    @R0,A
0228 E4 FA                      JMP     CLINST          ;JUMP TO CALL INSTRUCTION
```

```
 504                            ;
 505                            ;
 506                            ; * ERROR
 507                            ;
 508  022A  23 3F               ERROR:  MOV     A,#ERCR
 509  022C  14 3A                       CALL    COUT
 510  022E  14 94                       CALL    CRLF
 511  0230  23 08               ER05:   MOV     A,#08H
 512  0232  D7                          MOV     PSW,A
 513  0233  44 54                       JMP     PROMPT
 514                            ;
 515                            ;
 516                            ; * INIT
 517                            ;
 518                            ;
 519  0235  23 C0               INIT:   MOV     A,#SRST
 520  0237  3A                          OUTL    P2,A            ;RESET USART
 521  0238  23 41                       MOV     A,#ERST
 522  023A  3A                          OUTL    P2,A
 523  023B  23 CE                       MOV     A,#MODE         ;USART  MODE WORD
 524  023D  90                          MOVX    @R0,A           ;SEND USART MODE WORD
 525  023E  23 25                       MOV     A,#CMD          ;USART COMMAND WORD
 526  0240  90                          MOVX    @R0,A           ;SEND USART COMMAND WORD
 527  0241  9A FE                       ANL     P2,#0FEH
 528  0243  80                          MOVX    A,@R0           ;READ USART ONCE
 529                            ;
 530                            ;
 531                            ; * SIGNON
 532                            ;
 533                            ;
 534  0244  14 94               SGNON:  CALL    CRLF
 535  0246  14 94                       CALL    CRLF
 536  0248  23 20                       MOV     A,#SP
 537  024A  14 3A                       CALL    COUT
 538  024C  23 55                       MOV     A,#'U'
 539  024E  14 3A                       CALL    COUT
 540  0250  23 50                       MOV     A,#'P'
 541  0252  14 3A                       CALL    COUT
 542                            ;
 543                            ;
 544                            ; * PROMPT
 545                            ;
 546                            ;
 547  0254  14 94               PROMPT: CALL    CRLF
 548  0256  23 3A                       MOV     A,#PRMPT
 549  0258  14 3A                       CALL    COUT
```

```
551                                 ;
552                                 ;
553                                 ; * COMMAND
554                                 ;
555                                 ;
556  025A 14 2B                     CMD:    CALL    CIN
557  025C 14 3A                             CALL    COUT
558  025E AC                                MOV     R4,A
559  025F D3 53                             XRL     A,#'S'
560  0261 96 67                             JNZ     CMD05
561  0263 14 D6                             CALL    SCMD
562  0265 44 54                             JMP     PROMPT
563  0267 FC                        CMD05:  MOV     A,R4
564  0268 D3 52                             XRL     A,#'R'
565  026A 96 70                             JNZ     CMD10
566  026C 34 20                             CALL    RDCMD
567  026E 44 54                             JMP     PROMPT
568  0270 FC                        CMD10:  MOV     A,R4
569  0271 D3 57                             XRL     A,#'W'
570  0273 96 79                             JNZ     CMD15
571  0275 34 6C                             CALL    WRCMD
572  0277 44 54                             JMP     PROMPT
573  0279 FC                        CMD15:  MOV     A,R4
574  027A D3 47                             XRL     A,#'G'
575  027C 96 82                             JNZ     CMD20
576  027E 34 FA                             CALL    EXEC
577  0280 44 54                             JMP     PROMPT
578  0282 44 2A                     CMD20:  JMP     ERROR
```

```
                        ;
                        ;
                        ;       REQUEST BYTES
                        ;       0X - NO ACTION
                        ;       2N - REQUEST STATUS OF NODE N
                        ;       3N - CONNECT TO NODE N
                        ;       4X - DISCONNECT FROM NETWORK
                        ;
                        ;
                        ;       ACKNOWLEDGE BYTES
                        ;       0X - NO ACTION
                        ;       2P - STATUS ACK, NODE IS LISTENING TO NODE P
                        ;       3P - CONNECT TO NODE P
                        ;       4X - DISCONNECT
                        ;       5P - CONNECT NOT POSSIBLE, NODE BUSY
                        ;       6P - STATUS ACK, NO NODE
                        ;       7P - CONNECT NOT POSSIBLE, NO NODE
                        ;
                        ;
                        ; * CTA - CONNECT TO ARBITRATOR
                        ;
                        ;
0284  54 A2             CTA:    CALL    WFT
0286  9A BF                     ANL     P2,#0BFH        ;REQUEST ARBITRATOR
0288  56 88             CA05:   JT1     CA05            ;LOOP UNTIL ACKNOWLEDGE
028A  8A 40                     ORL     P2,#40H         ;RESET REQUEST
                        ;
028C  54 A2                     CALL    WFT             ;WAIT FOR END OF TRANSMISSION
028E  83                        RET
                        ;
                        ;
                        ;  *  CHKIN - CHECKS INPUT FROM USART
                        ;
                        ;
028F  9A F1             CHKIN:  ANL     P2,#USAB
0291  8A 01                     ORL     P2,#USOB
0293  80                        MOVX    A,@R0           ;READ USART STATUS
0294  53 02                     ANL     A,#RDRDY
0296  96 99                     JNZ     CIN5            ;JUMP IF CHAR IN
0298  83                        RET                     ;EXIT IF NO CHAR
0299  9A F0             CIN5:   ANL     P2,#UDAB
029B  80                        MOVX    A,@R0           ;READ CHARACTER
029C  83                        RET
```

```
                                ;
                                ;
                                ;  *  GNSP - GETS CHAR FROM INPUT, LOOPS UNTILL NON-SPACE
                                ;
                                ;
029D  54 8F                     GNSP:   CALL    CHKIN           ;CHECK INPUT
029F  C6 9D                             JZ      GNSP            ;LOOP IF NULL
02A1  83                                RET
                                ;
                                ;
                                ;  *  WFT -WAIT FOR END OF TRANSMISSION
                                ;
                                ;
02A2  9A F1                     WFT:    ANL     P2,#USAB
02A4  8A 01                             ORL     P2,#USOB
02A6  80                        WT05:   MOVX    A,@R0
02A7  53 04                             ANL     A,#TXE          ;MASK TRANSMITTER EMPTY
02A9  C6 A6                             JZ      WT05
02AB  83                                RET
                                ;
                                ;
                                ;  *  DCNCT - DISCONNECT NODE FROM NETWORK
                                ;
                                ;
02AC  54 84                     DCNCT:  CALL    CTA             ;REQUEST ARBITRATOR
02AE  23 40                     DCN05:  MOV     A,#40H          ;DISCONNECT BYTE
02B0  14 3A                             CALL    COUT
02B2  54 9D                             CALL    GNSP            ;GET RESPONSE
02B4  46 B4                     DCN10:  JNT1    DCN10           ;WAIT FOR END OF ACKNOWLEDGE
02B6  83                                RET
                                ;
                                ;
                                ;  *  CNNCT - CONNECT TO PORT IN REG R7
                                ;
                                ;
02B7  54 84                     CNNCT:  CALL    CTA             ;REQUEST ARBITRATOR
02B9  FF                                MOV     A,R7            ;GET REQUESTED PORT NUMBER
02BA  53 0F                             ANL     A,#0FH
02BC  43 30                             ORL     A,#30H          ;REQUEST TO CONNECT TAG
02BE  14 3A                             CALL    COUT            ;SEND REQUEST
02C0  54 9D                             CALL    GNSP            ;GET RESPONSE, WAIT FOR NON-NULL
02C2  AA                                MOV     R2,A            ;STORE RESPONSE IN R2
02C3  83                                RET                     ;RETURN TO USER ROUTINE
```

```
668                             ;
669                             ;
670                             ;   *  WFCNCT - WAIT FOR CONNECTION TO BE MADE
671                             ;
672                             ;
673  02C4  54 B7                WFCNCT: CALL    CNNCT           ;REQUEST ARBITRATOR AND SUBMIT REQUEST
674  02C6  53 F0                WFC5:   ANL     A,#0F0H         ;MASK OFF LOWER NIBBLE
675  02C8  D3 30                        XRL     A,#30H          ;COMPARE TO SUCCESSFUL ACK
676  02CA  96 CF                        JNZ     WFC10           ;JUMP IF NO SUCCESS
677  02CC  46 CC                WFC7:   JNT1    WFC7            ;WAIT FOR END OF ACKNOWLEDGE
678  02CE  83                           RET
679  02CF  54 AE                WFC10:  CALL    DCN05           ;GIVE TIME FOR OTHERS TO DCNCT
680  02D1  B8 50                        MOV     R0,#80
681  02D3  E8 D3                WFC15:  DJNZ    R0,WFC15
682  02D5  44 C4                        JMP     WFCNCT
683                             ;
684                             ;
685  02D7                               END
```

ASSEMBLER ERRORS = 0

SYMBOL TABLE

AD05	00A0	AD10	00A4	AD15	00AA	AD20	00AB
CA05	0288	CHKIN	028F	CIN	002B	CIN05	002F
CIN5	0299	CLINST	07FA	CMD	0025	CMD05	0267
CMD10	0270	CMD15	0279	CMD20	0282	CNNCT	02B7
COMD	025A	COUT	003A	COUT05	003F	CR	000D
CRLF	0094	CTA	0284	CTINT	07F4	DCN05	02AE
DCN10	02B4	DCNCT	02AC	ER05	0230	ERCR	003F
ERROR	022A	ERST	0041	EX05	0210	EX10	0212
EXEC	01FA	FI05	00BC	FILL	00AC	GNSP	029D
IB05	0085	INAD	009D	INBYTE	0076	INH05	0057
INH10	0063	INHEX	0049	INIT	0235	INTRPT	07F0
JTBL	000B	LF	000A	MODE	00CE	OFST	003E
OH05	0071	OUTBYT	0086	OUTHEX	0065	PRMPT	003A
PROMPT	0254	RC05	0133	RC10	013B	RC15	0153
RC20	0167	RC25	0169	RDCMD	0120	RDRDY	0002
SC05	00E3	SC10	00F1	SC15	0104	SC20	011F
SC25	00D4	SC30	011D	SCMD	00D6	SGNON	0244
SP	0020	SRST	00C0	TRRDY	0001	TXE	0004
UDAB	00F0	USAB	00F1	USOB	0001	WFC10	02CF
WFC15	02D3	WFC5	02C6	WFC7	02CC	WFCNCT	02C4
WFT	02A2	WR05	018D	WR10	01A5	WR15	01BB
WR20	01C0	WR25	01CE	WR30	01E5	WR35	01F8
WRCMD	016C	WT05	02A6	XREAD	00BD	XWRITE	00C8

APPENDIX B

SUBROUTINES IN 8748 MONITOR

There are a number of useful subroutines in the 8748 monitor that are accessable to the user. A jump table at the beginning of the monitor program allows direct entry to these subroutines. On the following pages are descriptions of the available routines and the entry and exit parameters as well as the calling procedures and register and port modification are listed.

The following is a list of the subroutines and a brief description of their purpose:

CHARIN	Input a single character from the console.
CHAROUT	Output a single character to the console.
CRLF	Output a carriage return and line feed.
OUTHEX	Convert four bits to hex and output.
INBYTE	Input two hex digits and pack into one byte.
OUTBYTE	Output one byte as two hex digits.
INAD	Input an address or data field.
XREAD	Read contents of external RAM.
XWRITE	Write into external RAM.
ERROR	Print error character and enter monitor.
PROMPT	Enter monitor.
CTA	Connect to arbitrator.
DCNCT	Disconnect from network.
CNNCT	Request connection to specified node.
WFCNCT	Same as CNNCT but wait for connect

CHARIN Input a single character from console

This routine will get a single character from the serial line and will leave the character in the accumulator. This routine will always wait for a character to be sent in through the USART.

ENTRY PARAMETERS - none

EXIT PARAMETERS - A reg = character received.

AFFECTED - output from P2

EXAMPLE:

```
CHARIN    EQU    000BH   ; location of routine
                         ;
          CALL   CHARIN  ; get character
                         ; character now in A reg.
```

CHAROUT Output a single character to the console

This routine will take the contents of the accumulator and send it out on the serial line. The routine will allways wait for the serial line to become ready for the written character.

ENTRY PARAMETERS - A reg = character to be output.

EXIT PARAMETERS - A reg = character output.

AFFECTED - R2, P2(lower)

EXAMPLE:

```
CHAROUT   EQU    000DH   ; location of routine
                         ;
          MOV    A,#20H  ; ASCII space character
          CALL   CHAROUT ; print space on console
```

CRLF Output a carriage return and line feed.

This routine will cause a carriage return and a line feed to be sent out on the serial port. This routine will wait for the serial line to become ready before sending the carriage return or line feed.

ENTRY PARAMETERS - none

EXIT PARAMETERS - none

AFFECTED - A, R2, P2(lower)

EXAMPLE:

```
CRLF    EQU     000FH   ; location of routine
                        ;
        CALL    CRLF    ; send to console
```

OUTHEX convert four hex bits to ASCII and output

This routine takes the lower four bits of the accumulator and converts them into an ASCII character and then calls CHAROUT to send this byte to the console. This routine waits for the serial line to become ready before the byte is output.

ENTRY PARAMETERS - A reg = lower four bits are number to be converted.

EXIT PARAMETERS - none

AFFECTED - R2, P2(lower)

EXAMPLE:

```
OUTHEX    EQU    0011H   ; location of routine
                         ;
          MOV    A,#02H  ; number two
          CALL   OUTHEX  ; convert to 032H and output
```

INBYTE Input two hex digits and pack into one byte

This routine will wait for two ASCII digits to be sent in through the serial port and convert each digit to a hex character and pack the two hex characters into one byte. If an ASCII character is input that cannot be converted, a flag is set and the exit parameter is no longer significant. This routine will also add the packed byte to the contents of register 7 and place this sum back into register 7. This addition is used for checksum calculation on hex file input.

ENTRY PARAMETERS - R7 - present checksum for hex file input.

EXIT PARAMETERS - A reg = packed byte
F0 = 0 when conversion valid
F0 = 1 when conversion invalid
R7 = updated checksum

AFFECTED - R2,R3,P2(lower)

EXAMPLE:

```
INBYTE  EQU     0013H   ; location of routine
                        ;
        MOV     R7,#0   ; clear checksum
        CALL    INBYTE  ; get two hex digits
        JF0     ERROR   ; error if non-valid
                        ; A reg is now packed byte
```

OUTBYTE Output one byte as two hex digits

This routine takes the accumulator and unpacks the upper and lower four bits, converts these two numbers into ASCII and outputs the two ASCII characters out through the serial port. The entry parameter is added to register 7 and the resulting sum is placed back into register 7 for checksum calculation.

ENTRY PARAMETERS - A reg = two hex numbers packed into one byte.
R7 = present checksum

EXIT PARAMETERS - R7 = updated checksum

AFFECTED - R2,R3,P2(lower)

EXAMPLE:

```
OUTBYTE   EQU    0015H    ; location of routine
                          ;
          MOV    R7,#0    ; clear checksum
          MOV    A,#5EH   ; byte to be output
          CALL   OUTBYTE  ; call routine
                          ; R7 = 5EH
                          ; a 35H and then a 45H
                          ; are sent out the serial
                          ; port
```

INAD Input an address or data field

This routine allows address specifications to be entered into the processor and may also be used for data substitution routines. This routine will allow numbers to be entered on the console and each number is converted to four bits and shifted into a twelve bit buffer. If a character that is not a valid hex number is entered, the routine returns control. The lower eight bits of the number are in R4 and the upper four bits are in R3. If a invalid hex caracter is entered on the first entry, then flag 0 will be set on return.

ENTRY PARAMETERS - none

EXIT PARAMETERS - R3 - upper four bits of address
R4 = lower eight bits of address
F0 = 0 when first character is not terminator
F0 = 1 when first character is terminator

AFFECTED - A reg, R2,R7

EXAMPLE:

```
INAD    EQU     0017H   ; location of routine
        CALL    INAD    ; invoke routine
        JF0     FIRST   ; if first character is
                        ; terminator
                        ; address is in R3:R4
```

XREAD Read contents of external RAM

This routine allows easy access to the external random access memory on the node microcomputer card. The address of the location to be read is in R6:R0 and the contents of this location is placed in the accumulator.

ENTRY PARAMETERS - R0 = lower eight bits of memory location.
R6 = upper four bits of memory location

EXIT PARAMETERS - A reg = contents of memory location

AFFECTED - P2(lower)

EXAMPLE:

```
XREAD   EQU   0019H   ; location of routine
                      ;
        MOV   R6,#04H
        MOV   R0,#00H ; address location 400H
        CALL  XREAD   ; invoke routine
                      ; A reg now contains
                      ; contents of location 400H
```

XWRITE Write into external RAM

This routine allows easy access to external random access memory. The address of the location desired is placed into R6:R0 and the desired contents should be placed in the accumulator before the call.

ENTRY PARAETERS - A reg = desired contents of memory location specified
R6 = upper four bits of memory address.
R0 = lower eight bits of memory address.

EXIT PARAMETERS - none

AFFECTED - R2,P2(lower)

EXAMPLE:

```
XWRITE    EQU    001BH   ; location of routine
                         ;
          MOV    R6,#04H
          MOV    R0,#00H ; memory address is 400H
          MOV    A,#5AH  ; desired contents are 5AH
          CALL   XWRITE  ; place 5AH in memory
                         ; location 400H
```

ERROR Print error character and enter monitor

This routine is the erroneous entry point to the monitor. The error character will be sent to the console and the execution will resume at the command prompt routine.

ENTRY PARAMETERS - none

EXIT PARAMETERS - no exit from routine

AFFECTED - irrelevant

EXAMPLE:

```
ERROR    EQU    001DH    ; location of routine
                         ;
         JMP    ERROR    ; enter monitor after
                         ; sending error character
                         ; to console
```

PROMPT Entry point for monitor

This location may be called when a program desires to terminate and give control to the command interpreter in the monitor. The prompt character is issued and the processor will wait for a command to be entered.

ENTRY PARAMETERS - none

EXIT PARAMETERS - no exit from routine

AFFECTED - irrelevant

EXAMPLE:

```
PROMPT    EQU    001FH   ; location of routine
                         ;
          JMP    PROMPT  ; reenter monitor
```

CTA Connect to arbitrator

This routine will cause the node executing to request communication with the arbitrator and will wait for an acknowledge before returning control to the calling program.

ENTRY PARAMETERS - none

EXIT PARAMETERS - none

AFFECTED - A reg, P2(lower)

EXAMPLE:

```
CTA     EQU     0021H   ; location of routine
                        ;
        CALL    CTA     ; node will be in
                        ; communications with
                        ; arbitrator processor
```

DCNCT Disconect from network

This routine forces the arbitrator processor to disconnect the node microcomputer from the network.

ENTRY PARAMETERS - none

EXIT PARAMETERS - none

AFFECTED - A reg, R2, P2(lower)

EXAMPLE:

```
DCNCT     EQU     23H     ; location of routine
                          ;
          CALL    DCNCT   ; disconnect node
                          ; from network
```

CNNCT Request connection to specified node

This routine allows inter-network connection to the node specified in register 7. The arbitrator's acknowledge to the request is returned in register 2.

ENTRY PARAMETERS - R7 = node to which connection is desired.

EXIT PARAMETERS - R2 = arbitrator acknowledge byte.

AFFECTED - A reg, P2(lower)

EXAMPLE:

```
CNNCT    EQU    0025H   ; location of routine
                        ;
         MOV    R7,#3   ; request to connect
                        ; to node number three
         CALL   CNNCT   ; invoke routine
                        ; if R2 = 83H, then
                        ; request was succsessfull
```

WFCNCT Same as CNNCT but wait for connect

This routine allows inter-network connection to the node specified in register 7. This routine will not return until the connection has been made. The user must check the status of the requested port before calling this routine, and if the port is not in the rack, this routine should not be invoked.

ENTRY PARAMETERS - R7 = node to which connection is desired

EXIT PARAMETERS - none

AFFECTED - A reg, R2, R0, P2(lower)

EXAMPLE:

```
WFCNCT    EQU    0029H   ; location of routine
                         ;
          MOV    R7,#3   ; request to connect
                         ; to node number three
          CALL   WFCNCT  ; invoke routine
                         ; now connected to
                         ; node number three
```

APPENDIX C

ARBITRATOR SOFTWARE LISTING

```
                        ;
0000                    ORG     0000H
4000            STACK   EQU     4000H           ;SYSTEM STACK
200C            RQP0    EQU     200CH           ;REQUEST PORT ZERO
200D            RQP1    EQU     200DH           ;REQUEST PORT ONE
2001            URTSTAT EQU     2001H           ;USART STATUS ADDRESS
2000            URTDATA EQU     2000H           ;USART DATA ADDRESS
0001            TRRDY   EQU     1               ;TRANSMITTER READY MASK
0004            TXEMPTY EQU     4               ;TRANSMITTER EMPTY MASK
0002            RDRDY   EQU     2               ;RECEIVER READY MASK
00CE            MODE    EQU     0CEH            ;USART MODE BYTE
0025            COMAND  EQU     25H             ;USART COMMAND BYTE
00FF            RSTAKB  EQU     0FFH            ;BYTE TO RESET ALL ACKNOWLEDGES
200C            ACKPTA  EQU     200CH           ;ACKNOWLEDGE PORT ADDRESS
200E            PRTNUM  EQU     200EH           ;ADDRESS OF ARBITRATOR PORT NUMBER
1000            DVPTB   EQU     1000H           ;DEVICE PORT BASE ADDRESS
200D            ARBPA   EQU     200DH           ;ARBITRATOR PORT ADDRESS
2007            CNTSTAT EQU     2007H           ;COUNTER TIMER STATUS ADDRESS
2006            CNTPARM EQU     2006H           ;COUNTER TIMER PARAMETER ADDRESS
00B6            CNTCNT  EQU     0B6H            ;COUNTER CONTROL WORD
0008            BDCNST  EQU     8               ;BAUD RATE CONSTANT
                        ;
                        ;INIT -
                        ;
0000  310040    INIT:   LXI     SP,STACK
0003  3ECE              MVI     A,MODE          ;SEND USART MODE WORD
0005  320120            STA     URTSTAT
0008  3E25              MVI     A,COMAND        ;SEND USART COMMAND WORD
000A  320120            STA     URTSTAT
000D  3A0020            LDA     URTDATA         ;READ USART ONCE
0010  210800            LXI     H,BDCNST        ;LOAD BAUD RATE CONSTANT INTO HL
0013  3EB6              MVI     A,CNTCNT        ;LOAD COUNTER CONTROL WORD TO A
0015  320720            STA     CNTSTAT
0018  7D                MOV     A,L
0019  320620            STA     CNTPARM         ;LOAD BAUD RATE CONSTANT
001C  7C                MOV     A,H
001D  320620            STA     CNTPARM
0020  3A0E20            LDA     PRTNUM
0023  320C20            STA     ACKPTA
0026  1E00              MVI     E,0
0028  CD6801    IN05:   CALL    STS             ;SET EACH DEVICE TO TALK TO SELF
002B  1C                INR     E               ;SET NEXT PORT NUMBER
002C  7B                MOV     A,E
002D  FE10              CPI     10H             ;COMPARE PORT TO 16
002F  C22800            JNZ     IN05            ;LOOP UNTIL ALL PORTS TALK TO SELF
```

```
                        ;
                        ;GNPR - GET NEW PROCESSOR REQUEST
                        ;
0032   3A0C20   GNPR:   LDA     RQP0            ;GET CURRENT REQUESTS FROM P0
0035   5F               MOV     E,A             ;STORE
0036   3A0D20           LDA     RQP1            ;GET REQUESTS FROM P1
0039   57               MOV     D,A
003A   010100           LXI     B,1
003D   78       GN05:   MOV     A,E             ;GET LSB OF REQUESTS
003E   A1               ANA     C               ;MASK OFF BIT
003F   CC6300           CZ      SPTA            ;IF PORT IS REQUESTING SATISFY REQUEST
0042   79               MOV     A,C             ;PUT MASK IN ACC
0043   81               ADD     C               ;MULTIPLY MASK BY TWO
0044   4F               MOV     C,A             ;RESTORE MASK
0045   3E08             MVI     A,8
0047   04               INR     B
0048   B8               CMP     B
0049   C23D00           JNZ     GN05            ;LOOP TO MASK OFF NEXT BIT
004C   010100           LXI     B,1
004F   7A       GN10:   MOV     A,D             ;GET MSB OF REQUESTS
0050   A1               ANA     C               ;MASK OFF BIT
0051   3E08             MVI     A,8
0053   CC6300           CZ      SPTA            ;IF PORT IS REQUESTING, SATISFY REQUEST
0056   79               MOV     A,C             ;MULTIPLY MASK BY TWO
0057   81               ADD     C
0058   4F               MOV     C,A
0059   3E08             MVI     A,8             ;
005B   04               INR     B
005C   B8               CMP     B
005D   C24F00           JNZ     GN10            ;LOOP TO MASK OFF NEXT BIT
0060   C33200           JMP     GNPR            ;GET NEW REQUESTS
```

```
                        ;
                        ;SPTA - SET PORT AND ARBITRATOR TO TALK TO EACH OTHER
                        ;
0063   E5       SPTA:   PUSH    H               ;SAVE PROCESSOR STATE
0064   D5               PUSH    D
0065   C5               PUSH    B
0066   F5               PUSH    PSW
0067   B7               ORA     A               ;SET FLAGS
0068   210C20           LXI     H,RQP0          ;GET ADDRESS OF REQUEST PORT 0
006B   CA7100           JZ      SP05            ;IF ENTERED WITH A=0, THEN JUMP
006E   210D20           LXI     H,RQP1          ;ELSE, GET ADDRESS OF REQUEST PORT 1
0071   7E       SP05:   MOV     A,M             ;GET CONTENTS OF REQUEST PORT TO A
0072   A1               ANA     C               ;COMPARE TO MASK IN C
0073   C21D01           JNZ     RSTACK          ;IF NOT ZERO, THEN PORT NOT REQUESTING
0076   F1               POP     PSW             ;RESTORE PREVIOUS AF
0077   F5               PUSH    PSW
0078   B0               ORA     B               ;GET REAL PORT NUMBER
0079   47               MOV     B,A             ;STORE REQUESTING PORT NUMBER IN B
007A   5F               MOV     E,A             ;AND IN E --
007B   CD5C01           CALL    RDPT            ;READ WHO PORT IS CONVERSING WITH
007E   E60F             ANI     0FH             ;MASK OFF LOWWER NIBBLE
0080   5F               MOV     E,A             ;STORE IN E
0081   CD6801           CALL    STS             ;SET OTHER PORT TO SELF
0084   58               MOV     E,B             ;RESTORE REQUESTING PORT NUMBER
0085   3A0E20           LDA     PRTNUM          ;PUT ARBITRATOR PORT NUMBER IN REGISTER A
0088   CD5001           CALL    STPT            ;SET PORT TO ARBITRATOR
008B   78               MOV     A,B             ;RESTORE REQUESTING PORT NUMBER
008C   320D20           STA     ARBPA           ;SET ARBITRATOR TO PORT
008F   320C20           STA     ACKPTA          ;ACKNOWLEDGE PORT
                        ;
                        ;GPRB - GETS PROCESSOR REQUEST BYTE
                        ;
0092   CD3901   GPRB:   CALL    CIN             ;READ CHARACTER FROM SERIAL PORT
0095   4F               MOV     C,A             ;STORE REQUEST WORD IN C
0096   E6F0             ANI     0F0H
0098   CA9200           JZ      GPRB            ;LOOP IF NULL CHARACTER
009B   79               MOV     A,C             ;RESTORE REQUEST WORD
009C   E6F0             ANI     0F0H
009E   FE30             CPI     030H            ;COMPARE TO CONNECT REQUEST
00A0   CAAB00           JZ      CONECT          ;CONNECT IF REQUEST TO CONNECT SET
00A3   FE20             CPI     020H            ;COMPARE TO STATUS REQUEST
00A5   CAF900           JZ      STATUS          ;INVESTIGATE STATUS
00A8   C3E600           JMP     DISCON          ;DISCONNECT IF ANY OTHER BYTE
```

```
                        ;
                        ;CONECT - SET TWO PROCESSORS TO TALK TO EACH OTHER
                        ;
00AB   79       CONECT: MOV     A,C
00AC   E60F             ANI     0FH             ;PUT REQUESTED PORT NUMBER IN A
00AE   4F               MOV     C,A             ;AND IN C
00AF   5F               MOV     E,A             ;AND IN E
00B0   CD5C01           CALL    RDPT            ;READ CONNECTION OF PORT REQUESTED
00B3   E610             ANI     10H             ;IS CARD IN RACK?
00B5   C2DF00           JNZ     NTINRK          ;IF CARD NOT PLUGGED IN, REQUEST NOT POSSIBLE
00B8   CD5C01           CALL    RDPT            ;READ CONNECTION OF PORT REQUESTED
00BB   E60F             ANI     0FH             ;MASK OFF TRASH
00BD   B9               CMP     C
00BE   C2D600           JNZ     PRTBSY          ;JUMP IF PORT NOT CONNECTED TO SELF
00C1   78               MOV     A,B             ;RESTORE REQUESTING PORT NUMBER
00C2   CD5001           CALL    STPT            ;SET REQUESTED PORT TO REQUESTING PORTS NUMBER
00C5   79               MOV     A,C             ;GET REQUESTED'S NUMBER TO A
00C6   F630             ORI     30H             ;SEND CONNECT POSIBLE
00C8   CD2B01           CALL    COUT            ;SEND REPLY
00CB   CD4501           CALL    WAIT            ;WAIT FOR END OF TRANSMIT
00CE   58               MOV     E,B             ;PUT REQUESTING PORTS NUMBER IN E
00CF   79               MOV     A,C             ;GET REQUESTEDS PORT NUMBER IN A
00D0   CD5001           CALL    STPT            ;SET REQUESTING PORT TO REQUESTED PORTS NUMBER
00D3   C31D01           JMP     RSTACK          ;RESET ACKNOWLEDGES
                        ;
                        ;PRTBSY - REQUESTED PORT IS BUSY
                        ;
00D6   79       PRTBSY: MOV     A,C             ;GET REQUESTED PORTS NUMBER
00D7   F650             ORI     050H            ;PUT REQUESTED PORT NUMBER IN PORT FIELD
00D9   CD2B01           CALL    COUT            ;SEND NO CONNECT-PORT BUSY
00DC   C39200           JMP     GPRB            ;GET AN OTHER PROCESSOR REQUEST BYTE
                        ;
                        ;NTINRK - REQUESTED PORT IS NOT IN RACK
                        ;
00DF   79       NTINRK: MOV     A,C             ;GET REQUESTED PORTS NUMBER
00E0   F670             ORI     070H            ;ASSEMBLE ACKNOWLEDGE
00E2   CD2B01           CALL    COUT            ;SEND NO CONNECT-NO PORT
00E5   C39200           JMP     GPRB            ;GET AN OTHER PROCESSOR REQUEST BYTE
                        ;
                        ;DISCON - DISCONNECT PROCESSOR FROM ANY ONE
                        ;
00E8   3E40     DISCON: MVI     A,40H
00EA   B0               ORA     B
00EB   CD2B01           CALL    COUT            ;SEND DISCONNECT ACKNOWLEDGE
00EE   CD4501           CALL    WAIT
00F1   78               MOV     A,B             ;PUT REQUESTING PORT NUMBER IN A
00F2   5F               MOV     E,A
00F3   CD6801           CALL    STS             ;CONNECT PORT TO ITSELF
00F6   C31D01           JMP     RSTACK          ;CLEAR ACKNOWLEDGES
```

```
                        ;
                        ;STATUS - SENDS STATUS OF PORT INDICATED TO PROCESSOR
                        ;
00F9   79       STATUS: MOV     A,C             ;GET PROCESSOR REQUEST BYTE
00FA   5F               MOV     E,A
00FB   CD5C01           CALL    RDPT            ;READ PORT INDICATED
00FE   E610             ANI     10H
0100   CA1001           JZ      ST05
0103   CD5C01           CALL    RDPT
0106   E60F             ANI     0FH
0108   F660             ORI     60H
010A   CD2B01           CALL    COUT
010D   C39200           JMP     GPRB
0110   CD5C01   ST05:   CALL    RDPT
0113   E60F             ANI     0FH             ;MASK LOWER NIBBLE
0115   F620             ORI     20H             ;OR IN STATUS ACKNOWLEDGE INFO
0117   CD2B01           CALL    COUT            ;SEND STATUS ACKNOWLEDGE
011A   C39200           JMP     GPRB            ;GET NEW PROCCESOR REQUEST BYTE
                        ;
                        ;RSTACK - CLEARS ALL ACKNOWLEDGES
                        ;
011D   3A0E20   RSTACK: LDA     PRTNUM
0120   320C20           STA     ACKPTA          ;ACKNOWLEDGE ARBITRATOR
0123   320D20           STA     ARBPA
0126   F1               POP     PSW             ;RESTORE PREVIOUS ENVIRONMENT
0127   C1               POP     B
0128   D1               POP     D
0129   E1               POP     H
012A   C9               RET
                        ;
                        ;COUT - SENDS CHARACTER TO SERIAL PORT
                        ;
012B   F5       COUT:   PUSH    PSW
012C   3A0120   C05:    LDA     URTSTAT
012F   E601             ANI     TRRDY
0131   CA2C01           JZ      C05
0134   F1               POP     PSW
0135   320020           STA     URTDATA
0138   C9               RET
                        ;
                        ;CIN - RECEIVES CHARACTER FROM SERIAL PORT
                        ;
0139   3A0120   CIN:    LDA     URTSTAT
013C   E602             ANI     RDRDY
013E   CA3901           JZ      CIN
0141   3A0020           LDA     URTDATA
0144   C9               RET
```

```
                        ;
                        ;WAIT - WAITS FOR END OF TRANSMISSION FROM USART
                        ;
0145   F5       WAIT:   PUSH    PSW
0146   3A0120   WO5:    LDA     URTSTAT
0149   E604             ANI     TXEMPTY
014B   CA4601           JZ      WO5
014E   F1               POP     PSW
014F   C9               RET
                        ;
                        ;STPT - SETS PORT(E) TO LISTEN TO PORT(A)
                        ;
0150   E5       STPT:   PUSH    H
0151   D5               PUSH    D
0152   1600             MVI     D,0
0154   210010           LXI     H,DVPTB
0157   19               DAD     D
0158   77               MOV     M,A
0159   D1               POP     D
015A   E1               POP     H
015B   C9               RET
                        ;
                        ;RDPT - READS PORT(E) INTO A
                        ;
015C   E5       RDPT:   PUSH    H
015D   D5               PUSH    D
015E   1600             MVI     D,0
0160   210010           LXI     H,DVPTB
0163   19               DAD     D
0164   7E               MOV     A,M
0165   D1               POP     D
0166   E1               POP     H
0167   C9               RET
                        ;
                        ;STS -  SETS PORT(E) TO SELF
                        ;
0168   F5       STS:    PUSH    PSW
0169   7B               MOV     A,E
016A   CD5001           CALL    STPT
016D   F1               POP     PSW
016E   C9               RET
                        ;
                        ;END -
                        ;
                        END
```

NO PROGRAM ERRORS

SYMBOL TABLE

* 01

A	0007	ACKPT	2000	ARBPA	200D	B	0000
BDCNS	0008	C	0001	CD6	012C	C|N	0130
CNTCN	0086	CNTPA	2006	CNTST	2007	COMAN	0025
CONEC	00AB	COUT	0128	D	0002	DISCO	00E8
DVPTB	1000	E	0003	GN05	003D	GN10	004F
GNPR	0032	GPRB	0092	H	0004	IN05	0028
INIT	0000 *	L	0005	M	0006	M80E	00CE
NTINR	000F	PRTBS	0006	PRTNU	200E	PSW	0006
RDPT	015C	RDRDY	0002	RDP0	200C	RDP1	200D
RSTAC	011D	RSTAK	00FE *	SP	0006	SP05	0071
SPTA	0063	STACK	4000	STATU	00F8	ST85	0110
STPT	0150	STS	0168	TRRDY	0001	TXEMP	0004
URTDA	2000	URTST	2001	W05	0146	WAIT	0145

REFERENCES

1. Intel Corporation, COMPONENT DATA CATALOG, 1980.

2. Intel Corporation, MCS-48 USER'S MANUAL, August, 1980.

3. Intel Corporation, MCS-85 USER'S MANUAL, September, 1978.

4. Texas Instruments Incorporated, THE TTL DATA BOOK, Second Edition, 1976.

5. Advanced Micro Devices, AM2900 FAMILY DATA MANUAL, 1978.