AN ABSTRACT OF THE THESIS OF

Ninan Mammen for the degree of Master of Science in Electrical and Computer Engineering presented on June 8, 1984.

Title: A Microprogramming Learning System

Redacted for privacy

Abstract Approved: ______________________________
Roy Rathja, Ph.D

A microprogramming teaching tool was designed and implemented. This tool was based on the Am 2900 bit-slice microprocessor family. It provides tools for understanding software development for a simple bit-slice microprocessor.

A Microprogramming Learning System

by

M. Ninan Mammen

A THESIS
submitted to
Oregon State University

in partial fulfillment of
the requirements for the degree of

Master of Science

Completed June 8, 1984

Commencement June 1985

APPROVED:

Redacted for privacy

Assistant Professor of Electrical Engineering in charge of major

Redacted for privacy

Head of department of Electrical Engineering

Redacted for privacy

Dean of Graduate School

Date thesis is presented June 8, 1984

Typed by Joanne Oshiro for Ninan Mammen

Acknowledgements

Thanks are due to Advanced Micro Devices, Incorporated, for their grant of the AM 2900 kit and the System 29. Their generosity made this project possible.

I would like to thank Don Heuster of Advanced Micro Devices, Incorporated for his invaluable help in obtaining the AM 2900 kit, and for continually putting me in contact with all the right people during the development stages of this work.

I would also like to thank Robert Searfus for being around to help with details about the System 29 and for his willingness to sit through and evaluate my oral presentation rehearsals.

Table of Contents

List of Figures

List of Tables

A MICROPROGRAMMING LEARNING SYSTEM

CHAPTER I

INTRODUCTION

Microprogramming is taught in senior level computer architecture classes at Oregon State University. In an effort to provide a practical example of microprogramming the design of a microprogramming teaching tool was undertaken.

To be useful in a lab situation, a microprogramming teaching tool must have the following characteristics:

1. Be simple to use.
2. Allow students to write and run their own microprograms.
3. Demonstrate the advantages and disadvantages of microprocessing.
4. Be practical to use to assign lab projects.

The Am 2900 Evaluation and Learning Kit was selected as the basis for the design of the microprogramming teaching tool. It had the following useful characteristics:

1. It was designed to demonstrate microprogramming techniques.
2. It could be used to load and run user microprograms.

---

[1]Donated to OSU by Advanced Micro Devices, Inc.

3. It was simple to modify if necessary.

The Am 2900 kit also had a number of disadvantages as follows:

1. It was tedious to use due to a slow and complicated program loading technique.
2. It was difficult to observe the flow of program execution due to a limited number of output indicator lights.

Thus while the kit was an excellent demonstration tool, it was not a practical lab instrument.

To make the kit useful for lab applications it was necessary to interface it with an external system. The external system would have to be able to simplify the program loading and data output procedures. Most mainframe, personal computers and development systems would be able to satisfy these requirements. The AMC System 29 was selected for the task. It was a microprogramming development system upon which future development in the microprogramming area was planned. It was also a system which would allow the greatest flexibility for expansion of the Am 2900 kit, beyond the modifications done in this project.

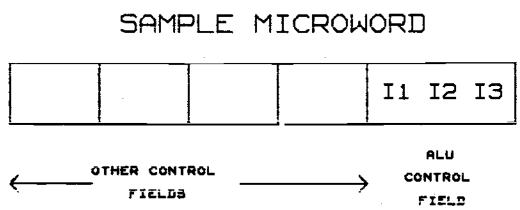
At the time of the conception of this project, it was decided to allow users of the combined system the option of using the AM 2900 kit independently or with the System 29.

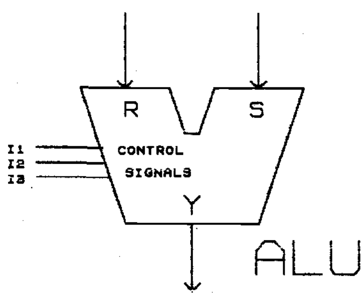
CHAPTER II

BACKGROUND INFORMATION

Microprogramming

The technique of microprogramming involves using microwords or microinstructions to run various devices within a microprocessor. A microword consists of a number of bits broken down into fields. Each field consists of the control bits required to run each device, an example of which is given in Figure 1. In the example a 010 (binary) in the ALU field would cause the ALU to do a S-R operation.





CONTROL SIGNALS

I1 I2 I3	ALU OPERATION
0 0 0	R + S
0 0 1	R − S
0 1 0	S − R
0 1 1	$R \vee S$
1 0 0	$R \wedge S$
1 0 1	$\overline{R} \wedge S$
1 1 0	$R \veebar S$
1 1 1	$\overline{R \vee S}$

Fig. 1. ALU Control.

A microprocessor with a large number of devices will require more fields and consequently a longer microword than a microprocessor with less devices. One of the fields within a microword is the next address field. This field is used with an address decoder or microsequencer to select the next microword to be executed. Microsequencers allow conditional and unconditional microword branching and subroutine calls. Microwords are stored in memory referred to as the microprogramming memory.

The Am 2900 Evaluation and Learning Kit

The Am 2900 kit is composed of an Am 2901 bit-slice microprocessor, an Am 2909 microprogram sequencer, registers, multiplexers (muxes) and several memories (see Figure 2). Each microprogram word on the kit consists of 32 bits which are broken down into 8 fields (see Figure 3). Sixteen microwords can be loaded and stored in the microprogram memory. These sixteen locations in memory are addressed by a four bit address from the Am 2909 sequencer.

Loading of the sixteen words is done through toggle switches which:

1. Select the address of the memory word to be loaded.

2. Set the data which is to be loaded.

3. Activate the load signal which writes the data into the memory.

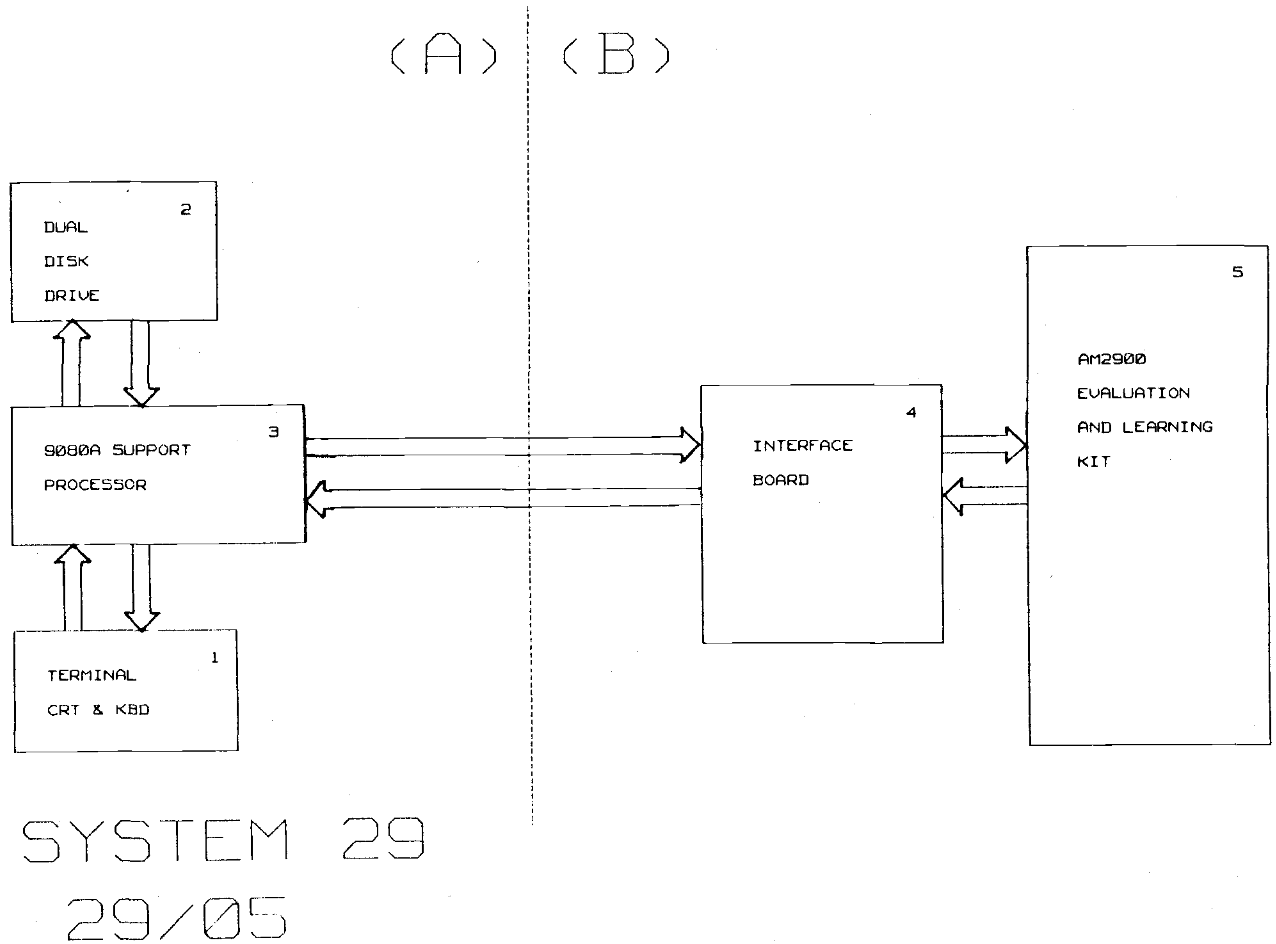


Fig. 2. Block diagram of Am 2900 Evaluation and Learning Kit.

RAM & MUX SELECT (FIELD NO.)	7	6	5	4	3	2	1	Ø
BIT NUMBER	31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-Ø
FIELD DEFINITION	BRANCH ADDRESS	NEXT INST. CONTROL	ALU DEST. CONTROL	ALU SOURCE SELECT	ALU CONTROL	A	B	D

Fig. 3. Am 2900 microprogram word.

Run modes are selected through toggle switches as well. A user may run the kit with the Single Step Switch or an external clock.

Operation of the kit may be observed from four indicator light emitting diodes (LED's). The LED's may be used to read outputs from both the 2901 and the 2909 as well as some of the other registers.

Two other sets of LED's allow the user to examine the microword memory and the pipeline register contents, (see Figure 2).

The pipeline register is a device used to speed up processing time by reducing the amount of time the system has to wait to fetch the next instruction. Microinstructions being executed are in the pipeline register. Thus while all the ALU functions and other micro-operations are being carried out, the next address can be decoded and the new microinstruction can be made ready at the inputs of the pipeline register. Thus when the present microinstruction has been executed, the next microinstruction has already been fetched and can be loaded and executed immediately.

Further details about microprogramming can be obtained from White, Bit-slice Design: Controllers and ALUs, and Sieworek, Bell, and Newell, Computer Structures: Principles and Examples and Mick and Brick Bit-slice Microprocessor Design.

Using the Am 2900 Kit

Once the user has developed a microprogram it can be loaded into memory using the three sets of input switches and a number of control switches (see Figure 2).

The RAM and Mux Select Switches. These three switches allow access to the eight different fields of the microprogram word. These switches are needed because the kit has only four data LED's, four pipeline LED's and four microprogram word LED's. Thus to view all 32 bits of each of the above registers it is necessary to view them four bits at a time. Since only four data switches are supplied it is also necessary to load the microprogram word in fields of four bits at a time. The RAM and Mux Select Switches allow the selection of these fields.

The Memory Address Switches. These four switches are used to address the sixteen microwords in the kit's microprogram memory.

The Memory Data Switches. These four switches[2] are set to the bit configuration that is to be loaded into the presently addressed microinstructions field[3] at the presently addressed memory location[4].

---

[2]The Memory Data Switches are set to opposite polarity due to the fact that the RAM used in the kit inverts data from input to output.

[3]selected using the RAM and Mux Switches

[4]selected using the Memory Address Switches

The Memory Load Switch. This momentary switch loads data into the microprogram memory.

The Run/ Load Switch. This toggle switch is used to select the mode of the processor. In Load mode, the data and address switches may be used to load (i.e. program) the RAM. In Run mode, the Memory Load Switch is inhibited and programs in memory can be run using the Single Step Clock Switch or the clock pulse input.

The Single Step Clock Switch. This momentary switch is used to apply a debounced clock pulse to all the clocked devices in the kit. This switch is also used to fill the pipeline register with the first microword. This is important because should the user attempt to run a program with random values in the pipeline register it will be impossible to guess which instruction will be fetched next. This switch is inhibited when the Single Step/ Pulse Generator Select Switch is set to Pulse Generator.

The Single Step/ Pulse Generator Select Switch. This toggle switch is used to decide if the kit will be run by the Single Step Clock Switch or an external pulse generator.

Summary. To use the kit the user sets the control toggle switches to Load mode and then stores a program into the 16 word RAM. With the address of the start location stored in the Memory Address Switches, the user fills the pipeline register with the first microinstruction to be executed. Finally the user runs the program in Run mode using a clock pulse or the Single Step Switch.

Further operation details may be obtained from The Am 2900 Learning and Evaluation Kit, User's Manual.

The System 29 Advanced Microprogramming Development System

The System 29 (29/05) consists of two main parts. The first part of the system is called the Microprogrammed System (MPS), the second part is called the System Support Processor (SSP). Both parts are housed in the System Mainframe.

The MPS consists of hardware required for microprogramming development. The MPS was not used in this project.

The SSP consists of a CPU card and a RAM card. The processor on the CPU card is an Am 9080A microprocessor which can run four RS232 serial I/O ports, three 8-bit parallel ports, a dual disk drive, a CRT, and various other peripherals. The CPU can also run the MPS part of the system if a user wishes.

Through the SSP, a user may run the entire system. CP/M is available to run on the Am 9080A. CP/M utilities include an Assembler, a Dynamic Debugging Technique routine and a Context Editor.

Additional System 29 details may be obtained from the AMDOS System 29 Users' Manual.

CHAPTER III

HARDWARE AND SOFTWARE CONSIDERATIONS AND DECISIONS

System Overview

Figure 4 is the block diagram of the main parts of the completed system. The area under (A) is the System 29 (29/05)[5] development system. It includes:

1. a terminal (CRT and keyboard)
2. a dual floppy disk drive
3. the 9080A system support processor.

The units under (B) include:

4. The interface between the Am 2900 and 29/05, which consists of muxes, latches, drivers, and receivers.
5. The Am 2900 kit which has been modified to run with the 29/05.

Am 2900 Kit Inputs

The 2900 kit, is set up with sets of switches which are used for addressing and data input. The interface board uses the switches on the kit and the outputs of the 29/05 as the two inputs to a bank of 2:1 multiplexers.

---

[5]Note System 29 and 29/05 are used interchangeably within this text.

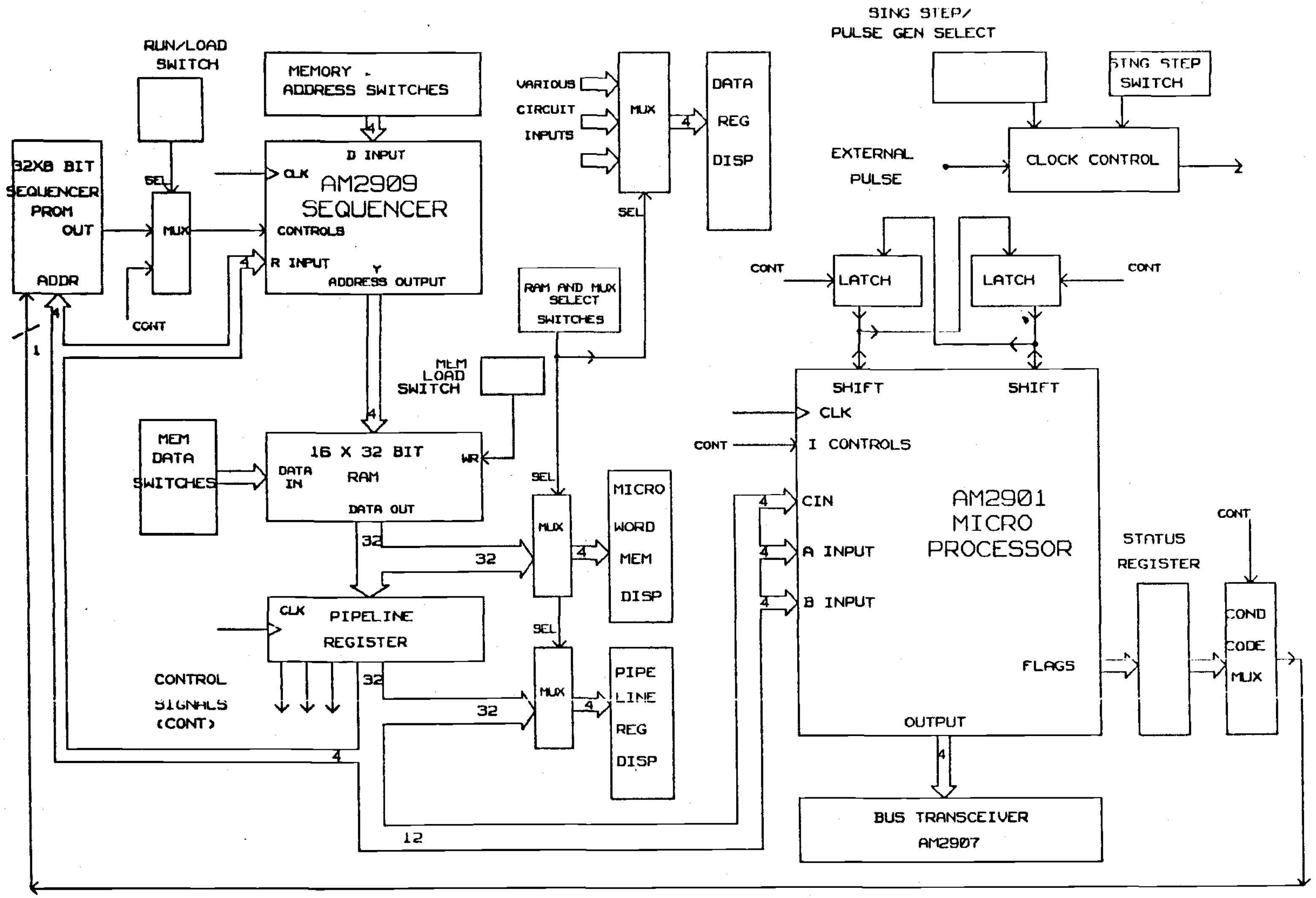


Fig. 4. Basic block diagram of the Am 2900 Microprogramming Learning System.

The multiplexer outputs then go to the inputs of the 2900 processor. This is shown diagrammatically in Figure 5. Thus a user can decide which inputs to use for the 2900 processor. A switch on the interface board allows the user to select the source he wishes.

An attempt was made to wire the system without cutting traces. However, after examining the Am 2900 circuitry, it was obvious that to make the system stable and reliable, traces would have to be cut before external connections could be made. A detailed description of these connections are given in the Appendix.

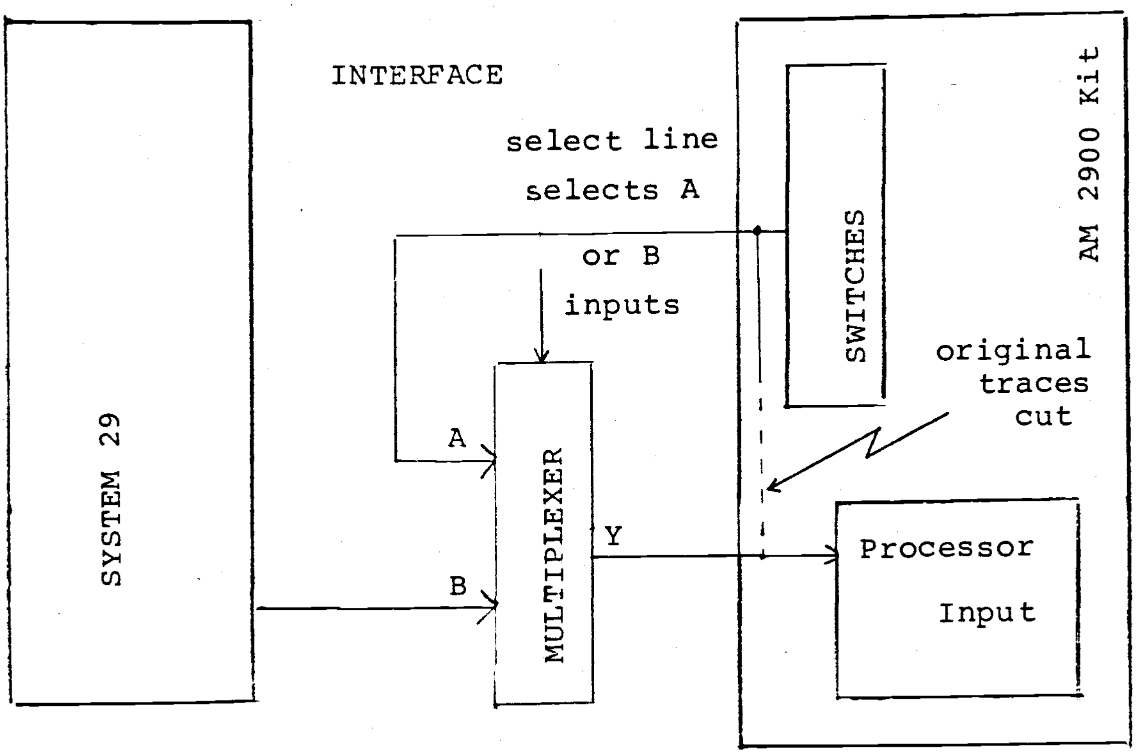


Fig. 5. Multiplexer input detail.

Am 2900 Kit Output

The method of output to the interface board was simple since it was possible to tap into the display outputs, just prior to the LED's of the 2900 kit as shown in Figure 6. Loading was not a concern due to the short distance the signals needed to be carried between the Am 2900 and the interface board.

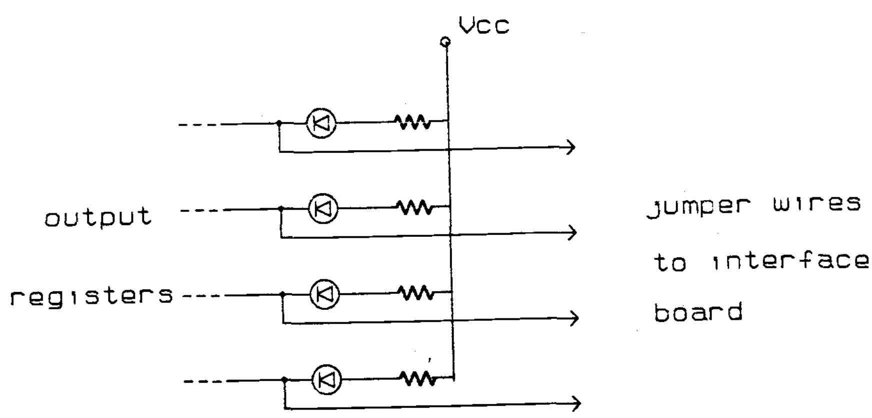


Fig. 6. Am 2900 output connection to Interface Board.

System 29 Input and Output

After the kit's method of output and input had been decided upon it was necessary to decide on the method of input and output on the 29/05. It was decided to use the three 8-bit bidirectional parallel ports on the 9080A processor board (see Figure 7).

Each of these ports can be configured as either inputs or outputs. Their configuration is decided by the following two factors:

1. How the ports are addressed from the 9080A, i.e. whether an OUT (port address) command or an IN (port address) is used.
2. Whether receivers or drivers are inserted into the optional sockets as shown in Figure 4.

Thus by putting drivers on all the ports that are to be outputs and using the OUT (port address) command, an output port configuration is determined. Likewise using receivers and the IN (port address) command an input port configuration is determined.

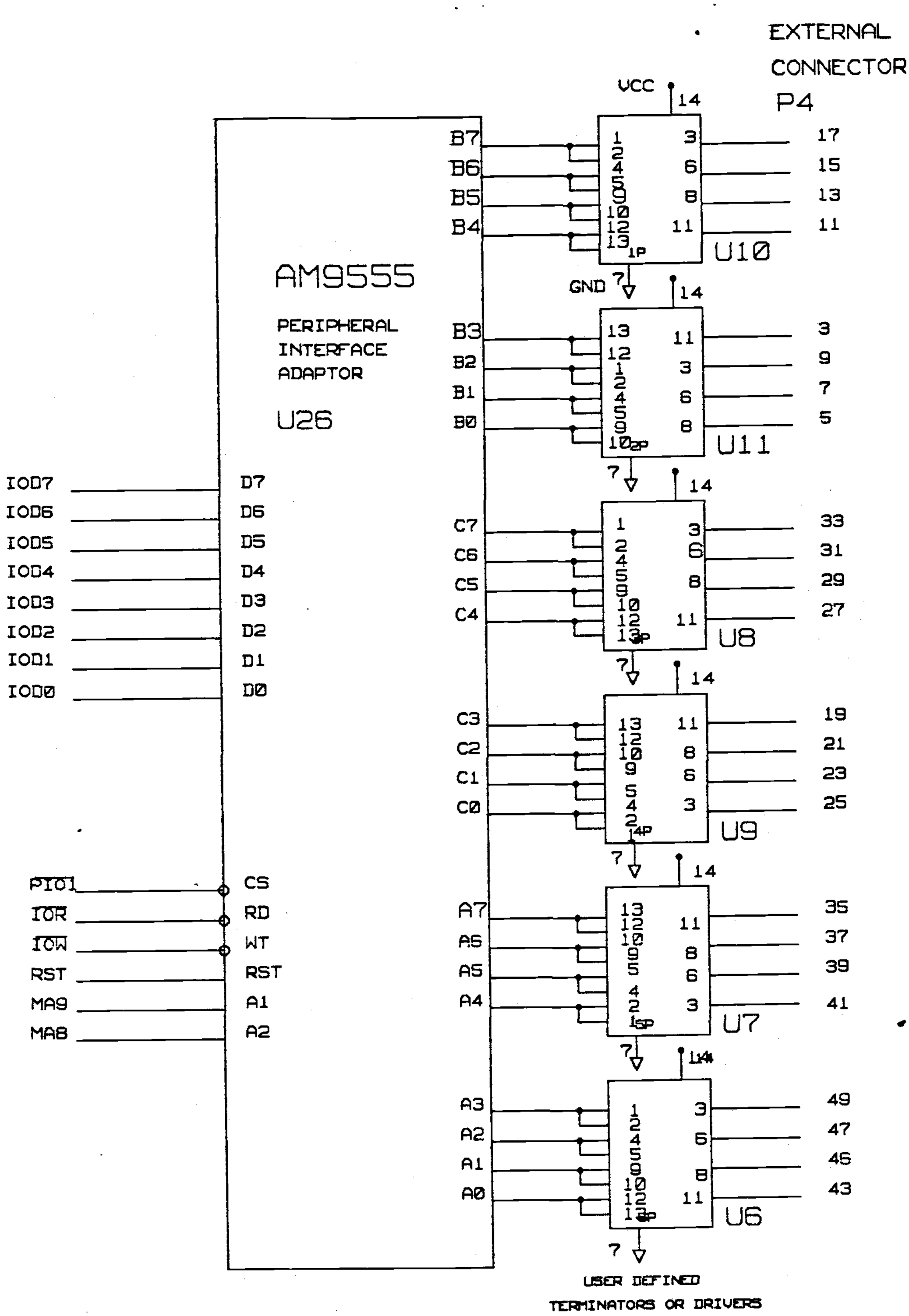


Fig. 7. System 29 parallel ports.

Interface Methods

There were two choices for running the interface between the two systems; the first was to run the system as it was intended to run, i.e. to use the 29/05 outputs to to simulate the 2900's switches. The other choice was to reconfigure the system and run it synchronously. The latter method would mean that the address would have to be present at a certain time 't'. At 't +τ' the data would be expected. Then once the data was present it could be loaded at the next pulse 't + 2τ '. This would of course involve timing restrictions, a clock pulse and a lot of extra timing logic. As a consequence it was decided instead to use the 29/05 to simulate the 2900 kit's switches. Also by using software to raise and lower the signal at one of the 29/05's output ports, the Single Step Clock Switch could be simulated. This avoided the necessity of using an external clock pulse to run the kit. This method though slower than the synchronous method is fast enough such that the time penalty is of no consequence and is unnoticed by the user.

Am 2900 Kit Load and Single Step Clock Pulses

The Load and Single Step Clock pulses on the 2900 are activated with momentary switches and each of these are used with a debouncer (an Am 9314). To avoid signal problems inputs to the interface board were taken by bypassing the debouncer as shown in Figure 8. In this way one is able to mux in and choose whichever inputs are desired to run the board (i.e. the kit's switches or the 29/05).

The 29/05 Bidirectional Port Drivers and Receivers

In deciding to use the 29/05's I/O ports, the selection of which ports to use as inputs and outputs had to be made. Furthermore, the method of driving the output ports and receiving the input signal had to be determined.

These I/O ports have configurations like NAND gates as shown in Figure 7. Due to the inability to locate drivers and receivers with the right configurations, the concept of an external adapter board for the 29/05 I/O was researched. This adaptor would consist of MC1488 drivers and MC1489 receivers. However this was discarded in favor of a simpler solution.

It was determined that regular TTL gates would have the capability to drive a 10-foot twisted pair cable needed to carry signals between the interface board and the 29/05. Thus 7400 NAND gates were used as drivers on the output ports and standard resistor pack were used for receivers on the input ports.

I/O Configurations and Connectors

Thus all the I/O and most of the format decisions

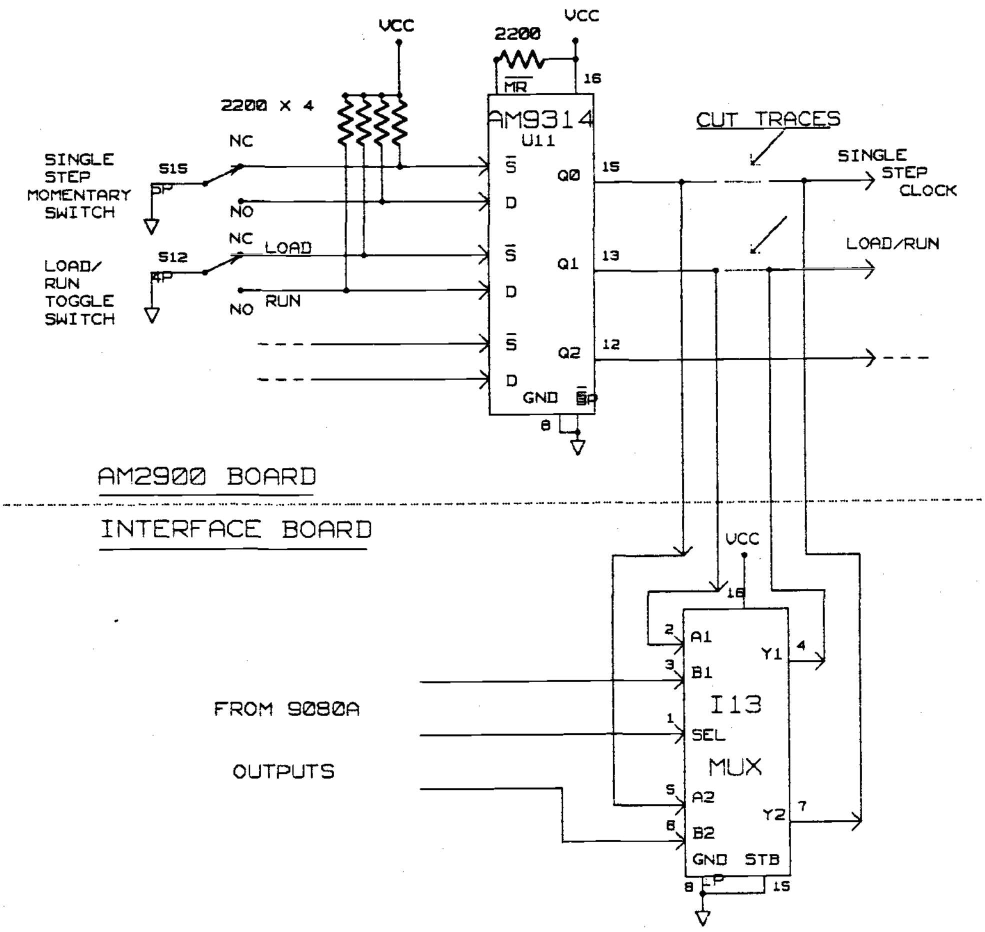


Fig. 8. Single step and load/run connections.

were made. What remained was the actual configuration of the I/O ports themselves, i.e. which ports would be used as outputs and which as inputs. To decide on a configuration it was necessary to determine how the signals were going to be routed to the 10 foot cable. The 29/05 uses a 50 pin edge connector and ribbon cable to route the 24 parallel port I/O signals (3 ports X 8 bits) to a DB25 pin connector on the frame. Every alternate line on the edge connector is grounded. Unfortunately a lot of the DB25 pins are also grounded, and only port A and two lines of port C are actually connected to the DB25 connector. The other signal lines were unconnected. By determining which connector pins were unused and by using jumper wires to connect the I/O port lines to these pins, all necessary signals were routed to the connector without any traces being cut.

Am 2900 Kit I/O Requirements

The kit requires:

a. 4 data inputs

b. 4 address inputs

c. 3 Mux/ RAM select inputs

d. 1 Single Step Clock Pulse signal

e. 1 Memory Load signal

f. 1 Load/ Run selection input

g. 1 Clock Pulse/ Single Step selection input

h. 4 data outputs

i. 4 pipeline register outputs

j. 4 microword memory register outputs

The above totals 15 input signals and 12 output signals. However, if the data needed can be latched in at the 2900 kit's input, all the lines will not have to be used simultaneously. This method is shown in Figure 9, the Interface Block Diagram. This bus-like structure enables each I/O line to be used for more than one input signal.

Am 29/05 Port Configurations

The 29/05 ports were broken down into input and output as follows:

Port A -- output from 29/05 to 2900 used for Data, Address, etc.

Port B -- output from 29/05 to 2900, used for control signals, clock pulses to latches, memory loads, etc.

Port C -- input to 29/05 from 2900, used for reading data register contents, pipeline register contents, and memory contents.

Rather than using all eight output lines of Port B, Port B was used to run a 3-8 decoder (a 74LS138). This required only three lines from Port B(B_0 to B_2). By using Port B as a control port, data from Port A can be clocked into the desired latches.

Through the decoder, Port B is also used to run the

Single Step Clock Pulse and the Memory Load Pulse[6].

The three output registers are muxed[7] to the lowest significant nibble of Port C. The selection of sources for the mux is made with lines B_5 and B_4 of Port B.

This resulted in the input/output configuration shown in Table 3.1 and the block diagram shown in Figure 9.

The Interface Board

The interface board was wired using wirewrap methods, that being the easiest to debug and modify. The board was designed to be as physically modular as possible, i.e. it can be disconnected from the 2900 kit for testing or modifying. This also allows the entire system to be moved without breaking wires or loosening connections.

A schematic of all connections is included in the Appendix.

Software Decisions

Having decided on the hardware options, the software

---

[6] The Single Step/ Pulse Generator Switch does not need to be accessed by the 29/05. The user must make sure it remains in the Single Step position whenever using the simulation program.

[7] By using a mux to select a source it was possible to reduce the number of 29/05 input lines needed and thus use the DB25 connector with its limited number of pins.

Table 3.1. Table of I/O signals.

Inputs to Interface Board from 9080

2900 Signal Name	29/05 Signal Name	Edge Conn.	DB25 Pin No.	Jumper Pin No.	Color
GND	GND	1	P_{15}	J4-2	Red
0	A_0	43	P_1	J4-3	Blue
DATA BUS 1	A_1	45	P_2	J4-4	Grn
DATA BUS 2	A_2	47	P_3	J4-5	Org
DATA BUS 3	A_3	49	P_4	J4-6	Yellow
DATA BUS 4	A_4	41	P_5	J4-7	Brown
DATA BUS 5	A_5	39	P_6	J4-8	Violet
DATA BUS 6	A_6	37	P_7	J4-9	w/Blk
DATA BUS 7	A_7	35	P_8	unused	
CONTROL 0	B_0	5	P_{18}	J4-11	w/Blue
CONTROL 1	B_1	7	P_{19}	J4-12	w/Grn
CONTROL 2	B_2	9	P_{20}	J4-13	w/Org
UNUSED	B_3	3	P_{21}	unused	
MUX SELECT 1	B_4	11	P_{22}	J4-15	w/Brn
MUX SELECT 2	B_5	13	P_{23}	J4-16	w/Vio
UNUSED	B_6	15	N/C	unused	
UNUSED	B_7	17	N/C	unused	

Outputs from Interface to 9080

OUTPUT BUS 0	C_0	25	P_{13}	J5-1	Blk
OUTPUT BUS 1	C_1	23	P_{24}	J5-2	Red
OUTPUT BUS 2	C_2	21	P_{25}	J5-3	Blue
OUTPUT BUS 3	C_3	19	P_{14}	J5-4	Grn
UNUSED	C_7	33	P_{17}	N/C	
GND	GND		P_9	GND	
GND	GND		P_{10}	GND	
GND	GND		P_{11}	GND	
GND	GND		P_{12}	GND	
GND	GND		P_{16}	GND	

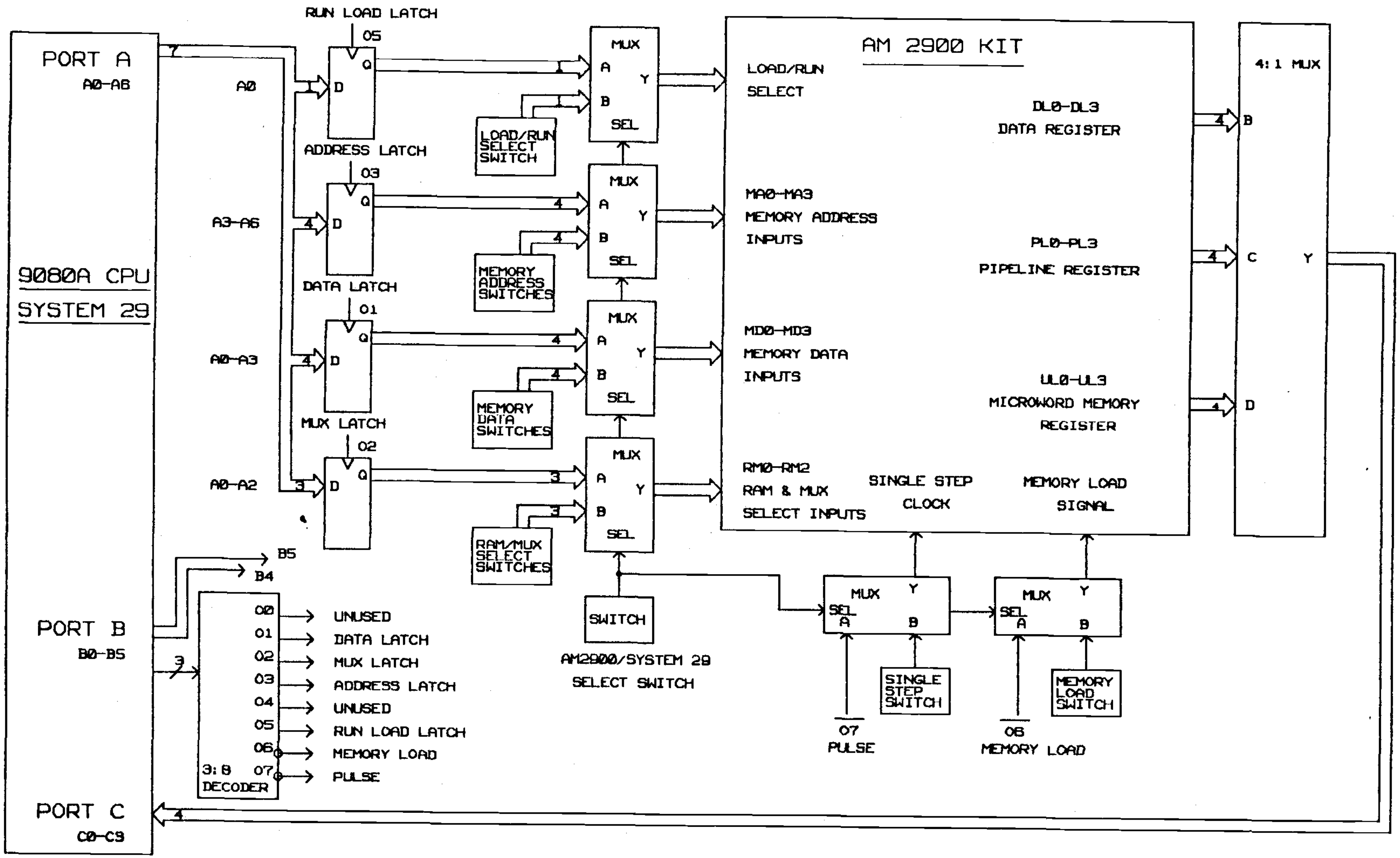


Fig. 9. Interface Block Diagram.

decisions had to be made.

The main subroutines for each operation, e.g. loading, reading, etc. are simple and are explained below.

Loading a nibble of data into memory

Below is a detailed example of a microprogram memory load subroutine. It is suggested that the reader follow these steps while referring to Figure 9, the Interface Block Diagram.

This subroutine loads one nibble (field) of a microword at a time. Field 3 of the microword[8] is used for this example, i.e. the ALU function and the Am 2901 carry input.

A few assumptions are made in this example:

1. The address of the microword is located in the register called ADDR.
2. The data to be loaded is located in the register addressed as DATA.

---

[8]Selected by using the RAM and mux select as listed on page 3-4 of the Am 2900 manual.

Assembly Listing of Program

```
LOAD$NIBBLE:

MVI A, 01H     ; Put the 2900 into Load mode by sending
OUT PORT$A     ; out a 1 to Port A (bit A0) and
               ; then latch this into the 1 bit latch.
MVI A, 05H     ; This is done by sending out a
               ; 10 (binary) to Port B which
OUT PORT$B     ; will send a high signal out
               ; line O5 (Run/ Load Latch).
MVI A, 00H     ; The line O5 is lowered to simulate a
OUT PORT$B     ; single clock pulse.

LDA ADDR       ; Load in the 4 bit address.
RLC            ; Move the address to bits A6-A3 as
RLC            ; these are the output lines connected to
               ; the address mux on the
RLC            ; interface board.
ORI 03H        ; OR in the RAM/ Mux Select choice.
               ; This results in a 7 bit address as
               ; shown in Figure 10.
OUT PORT$A     ; now send this address out PORT$A,
MVI A,03H      ; latch in the Address and then latch in
OUT PORT$B     ; the RAM/ Mux Select. This is done
               ; using lines O3 and O2 respectively.
MVI A, 02H     ;
OUT PORT$B     ;

LDA DATA       ; Send
OUT PORT$A     ; the data out port A in bits A0-A3.

MVI A, 01H     ; And in the same way latch the data in
OUT PORT$B     ; using the line O1.
MVI A,06H      ; Now that the address and data are in
OUT PORT$B     ; send out a memory load pulse. This is
MVI A, 00H     ; done using O6.
OUT PORT$B     ; Lower the signal to simulate a single
               ; pulse.
```

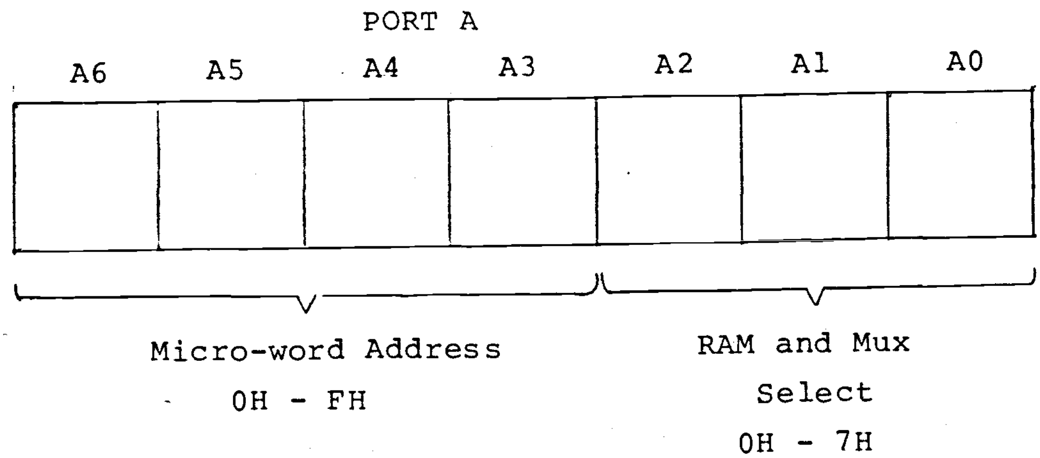


Fig. 10. Effective 7 bit address for a microword field.

Thus the microword can be loaded for all 8 RAM and mux locations (from 0 to 7).

Reading a nibble of data

To read the Am 2900 kit's outputs it is necessary to select which register to read with the control signals of Port B see Table 3.2 below. Then using the IN PORTØC command on the 9080A the 4 bit data can be loaded through Port C into the LSN of the 9080A accumulator (see Figure 9).

Table 3.2. Control Signals for Port C source.

B_5 B_4	Data on Port C bits C_0-C_4
0 0	N/C
0 1	Data Register D_0-D_3
1 0	Pipeline Register P_0-P_3
1 1	Microword Memory Register M_0-M_3

Running a program

In running a program on the 2900 kit the kit is first put into run mode and line O_7 is used as a Single Step Clock input. Software is used to create the pulse.

The Simulation Monitor

The simulation monitor is the software that controls the system and simulates the Am 2900 kit switches. All user input and output through the terminal is handled by the monitor using CP/M utilities. The user's input is decoded by the monitor and the appropriate load, read or run subroutine is executed.

All software is written in assembly language and is listed in the Appendix.

CHAPTER IV

USING THE AM 2900 MICROPROGRAMMING LEARNING SYSTEM

Using the Am 2900 with the 29/05

To use the simulation program:

1. Turn on the power switches on the CRT, disk drive, processor and Am 2900 kit.
2. Set the Am 2900 Single Step/ Pulse Generator Select Switch to Single Step.
3. Set the 29/05 - Am 2900 Select Switch on the Interface Board to 29/05.
4. Hit the reset button on the 29/05 processor.
5. Insert the "Am 2900 Simulation" disk into Drive A.

The system will boot up with a request for the date. Once this has been entered the system will return with a CP/M prompt:

A>

The user then types in the simulation program name and enters the simulation monitor[9].

A> **AM2900 <cr>**

---

[9]For clarity, input typed by the user is shown in bold face type with <cr> representing a carriage return.

The system responds with the monitor title and the simulation prompt:

-

and the user may then enter any command as follows.

a) Loading microword with data. The L (Load) command:

```
-L3 <cr>                       / L for Load, 3 for
                               / location 3 onwards.
3-AF011001  56A23101 <cr>      / System shows original
                               / data, user enters new
                               / data.
4-ACE611AF <space> <cr>        / Old data is left
                               / unchanged.
5-4653ABCD <cr>                / Exit Load mode.

-                              / System prompt.
```

The microword is displayed highest significant nibble first. The user will need to refer to the Am 2900 user's manual for microprogramming details.

b) Reading registers and data. The X (Examine) command.

```
-XD <cr>                       / Examine data register.

D-ABCDEF12                     / System responds with
                               / the 8 data nibbles.
                               / Note the lowest
                               / significant nibble is
                               / the currently addressed
                               / memory location.
-XM <cr>                       / Examine presently
                               / addressed microword.
M-12345678                     / System responds with
                               / 8 microword nibbles.
-XP <cr>                       / Examine present
                               / pipeline register
                               / contents.
P-54321ACE                     / System responds with 8
                               / pipeline nibbles.
```

```
-X <cr>                                   / Examine all output
                                          / registers

D-12345678      P-9ABCDEF0      M-1A2B3C4D

                                          / System responds with
                                          / contents of all three
                                          / output registers.
-                                         / System prompt.
```

c) Running the program, the G (Go) command[10].

```
-G3,F <cr>                              / G for go, 3 for start
                                        / location F for
                                        /  breakpoint[11]
3=D-54002192    P-21000000    M-F1000000
2=D-AA65430F    P-F1000000    M-F1000000
```

System responds by running program from location 3 outputting address of the location, 8 data nibbles, 8 microword nibbles and 8 pipeline register nibbles at each step (most significant nibble first). User can break program at any time by hitting any key on the console.

d) The S (Single Step) command[10].

```
-S2 <cr>                                / S for single step,
                                        / 2 for start location.
2=D-ABCDEFO     P-18675309     M-54002192
```

System responds by running program from location 2 outputting as above the address of the location, 8 data

---

[10] The G and S commands are executed in Run mode. Should the user wish to execute commands in the Load mode (certain examples in the Am 2900 kit user's manual do this) it is suggested that he do so using the Am 2900 kit independently of the 29/05.

[11] The microword at the breakpoint is not executed. The contents of the output registers at this point may be viewed by using the X command.

nibbles, 8 microword nibbles and 8 pipeline register nibbles (most significant nibble first). User continues single stepping by hitting Carriage Return. To exit single step mode the user hits any other key before Carriage Return.

e) The E (Exit) command.

-E <cr>	/ E for Exit
A>	/ System returns to CP/M

f) Errors -- any undefined inputs will cause the system to respond with a bell and a '?'. The system then returns with a prompt for the next command.

Running the Am 2900 Kit Independently of the 29/05

Power up the 2900 kit and the interface board, set Am 2900 - 29/05 switch to Am 2900. Now the switches on the kit will be used as the input to the Am 2900 kit with no interference from the 29/05 development system.

CHAPTER V

CONCLUSION AND FUTURE EXPANSION SUGGESTIONS

The Am 2900 kit is an effective bit-slice demonstration utensil, yet the author feels that AMD should have added external I/O capabilities. This could be in the form of a bus, which would enable the kit to be used for more than just demonstration. Had these interface options been present, kit users would have been able to see some real world applications of bit-slice.

The Am 2900 kit is already very useful for understanding the intricacies of microprogramming. However, it is very tedious to program and if it were to be used as a teaching tool, students could easily get confused by the technicalities of its use. The end effect is that more time is spent wrestling with the methods of loading a program than the actual details of microprogramming.

The objective of this work was to interface the kit with a development system and make a system that would be simple to program, and practical to use for class projects. This goal has been achieved and the modified kit proved to be more versatile and convenient to use. Students may now load, debug, run and most importantly watch the flow of an executing program while on line. Observing the flow of an executing program is vital to the total understanding of any system, especially one as complex as a microprogrammed microprocessor. Prior to the modifications, a student would have found this almost impossible to do, due to the limited display capabilities of the kit.

Much can be done to extend the present project and all software has been written to allow for the ease of modification or extension.

The following are a few possible ideas for future students:

1. Write software that will break down the 2900 microword into its separate fields and actually list what action is taking place. The user will then be able to decode and debug the micro-word while on-line.

2. Write software that will allow 2900 programs of longer than 16 microwords. What is envisioned is that the user writes in a program of any length. The 29/05 then loads a page of 16 words. When the page of microinstructions has been executed the 29/05 will automatically load in the next page and so on until the entire program has been run. Users will be limited to branching within the current page, unless a system of branching outside each page is developed. This would not be too complicated as all that would be needed is

to add an extra nibble[12] to each microword which would be the branch page number. On every branch instruction the 29/05 then halts Am 2900 execution and checks if the branch is in the current page or not. If it is not in the current page the 29/05 loads the branch page next and continues processing from there.

3. Add more data and pipeline LED's on the interface board so that a user may see all 32 data pipeline or memory word bits at a single glance.

4. Use the bit-slice capabilities of the Am 2901 to expand the ALU length of the Am 2900 kit to an 8 or 16 bit microprocessor.

---

[12]This nibble would not be loaded into the Am 2900 microword memory but would be stored and used by the 29/05 for branching.

BIBLIOGRAPY

1. "Am 2900 Learning and Evaluation Kit, User's Manual." Advanced Micro Devices, Inc., 1980.

2. "AMDOS System/ 29 Manual." Advanced Micro Devices, Inc., 1978.

3. White, Donnamaie E. Bit-Slice Design: Controllers and ALUs. Garland and STPM Press, 1981.

4. Sieworek, Bell and Newell. Computer Structures: Principles and Examples. McGraw-Hill. 1982. Chapters 11, 12, 13 and 14.

5. Mick and Brick. Bit-Slice Microprocessor Design. McGraw-Hill. 1980.

Appendix

Table A1. Drivers* and Terminators used for 9080 parallel ports.

Port Name and bits	Configuration	Socket Number	Chip Type	Chip No.
A_{0-3}	OUTPUT	V_6	NAND	7400
A_{4-7}	OUTPUT	V_7	NAND	7400
B_{0-3}	OUTPUT	V_{11}	NAND	7400
B_{4-7}	OUTPUT	V_{10}	AND	7408
C_{0-3}	INPUT	V_9	Resistor pack	221/331
C_{4-7}	UNUSED	V_8	--	--

Table A2. Jumper color coding/ pin outs

Pin Number	Color
1	Black
2	Red
3	Blue
4	Green
5	Orange
6	Yellow
7	Brown
8	Violet
9	White/ Black
10	White/ Red
11	White/ Blue
12	White/ Green
13	White/ Orange
14	White/ Yellow
15	White/ Brown
16	White/ Violet

* refer to Figure 4.

Table A3. Jumper pin outs.

Signals from Am 2900 kit to Interface

Jumper pin	From Switch No.	Am 2900 Signal Name
J1-1	S_{11}	MAS_3
J1-2	S_{10}	MAS_2
J1-3	S_9	MAS_1
J1-4	S_8	MAS_0
J1-5	S_4	MDS_3
J1-6	S_5	MDS_2
J1-7	S_6	MDS_1
J1-8	S_7	MDS_0
J1-9	S_3	RMS_2
J1-10	S_2	RMS_1
J1-11	S_1	RMS_0
J1-12	Q_0	single step clock switch input
J1-13	Q_1	load/ run switch input
J1-14	S_{14}	memory load switch input
J3-1		DL_3
J3-2		DL_2
J3-3		DL_1
J3-4		DL_0
J3-5		PL_3
J3-6		PL_2
J3-7		PL_1
J3-8		PL_0
J3-9		UL_3
J3-10		UL_2
J3-11		UL_1
J3-12		UL_0
J3-15		GND
J3-16		+5

Table A3. Jumper pin outs, cont.

Signals from Interface to Am 2900 kit

Jumper Pin	Am 2900 Kit Signal Name
J2-1	MA_3
J2-2	MA_2
J2-3	MA_1
J2-4	MA_0
J2-5	MD_3
J2-6	MD_2
J2-7	MD_1
J2-8	MD_0
J2-9	RM_2
J2-10	RM_1
J2-11	RM_0
J2-12	single step clock signal
J2-13	load/ run select signal
J2-14	memory load signal
J2-15	GND
J2-16	GND
J1-15	Vcc
J1-16	Vcc

Table A4. Inputs from 9080 to Interface Board

DB25	to Jumper Pin	Color	Signal Name	Edge Conn.
P1	J4-3	Blue	A_0	43
P2	J4-4	Green	A_1	45
P3	J4-5	Orange	A_2	47
P4	J4-6	Yellow	A_3	49
P5	J4-7	Brown	A_4	41
P6	J4-8	Violet	A_5	39
P7	J4-9	W/Black	A_6	37
P8	unused		A_7	35
P9	unused		GND	
P10	unused		GND	
P11	unused		GND	
P12	unused		GND	
P13	J5-1	Black	C_0	25
P14	J5-4	Green	C_3	19
P15	J4-2	Red	GND	
P16	unused		GND	
P17	unused		C_7	33
P18	J4-11	W/Blue	B_0	5
P19	J4-12	W/Green	B_1	7
P20	J4-13	W/Orange	B_2	9
P21	unused		B_3	3
P22	J4-15	W/Brown	B_4	11
P23	J4-16	W/Violet	B_5	13
P24	J5-2	Red	C_1	23
P25	J5-3	Blue	C_2	21

Table A5. Table of Parts.

Chip No.	Chip Name	Function
I1	74LS138	1 of 8 decoder/ demux
I2	74LS 04	Inverter
I3	74LS175	Quad D Latch
I4	74LS175	Quad D Latch
I5	74LS175	Quad D Latch
I6	74 157	Quad 2:1 Mux
I7	74 157	Quad 2:1 Mux
I8	74 157	Quad 2:1 Mux
I9	74LS175	Quad D Latch
I10	74LS253	Dual 4:1 Mux
I11	74LS253	Dual 4:1 Mux
I12	74 00	Two Input NAND
I13	74 157	Quad 2:1 Mux
I14	74 00	Two Input NAND
I15	74 00	Two Input NAND
I16	74 00	Two Input NAND

Table A6. Jumper Cables

J1	From 2900 to Interface Board
J2	From Interface Board to 2900
J3	From 2900 to Interface Board
J4	From System (9080) to Interface Board
J5	From Interface Board to System (9080)

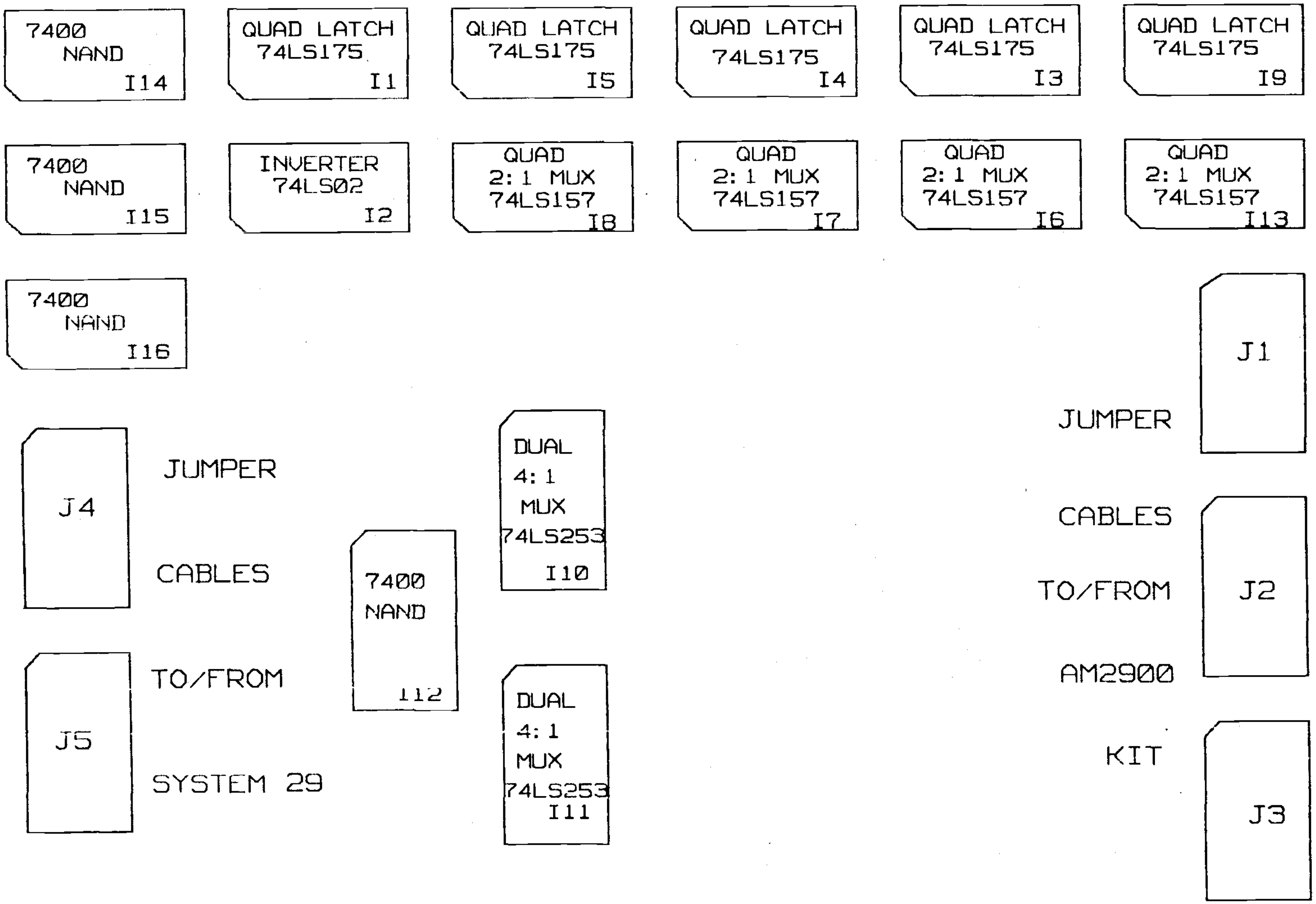


Fig. A1. Interface board layout: top view.

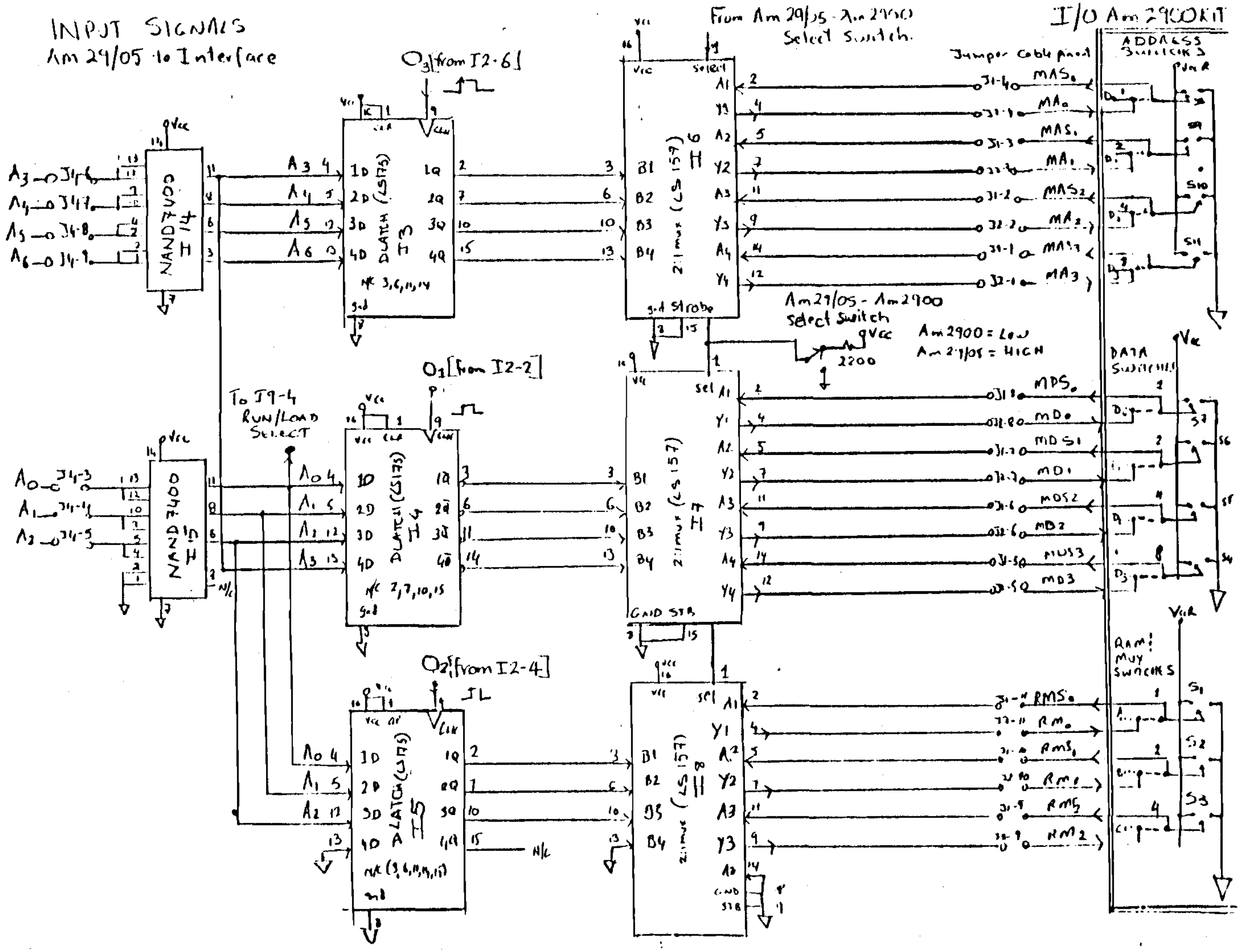


Fig. A2. Interface board schematics I.

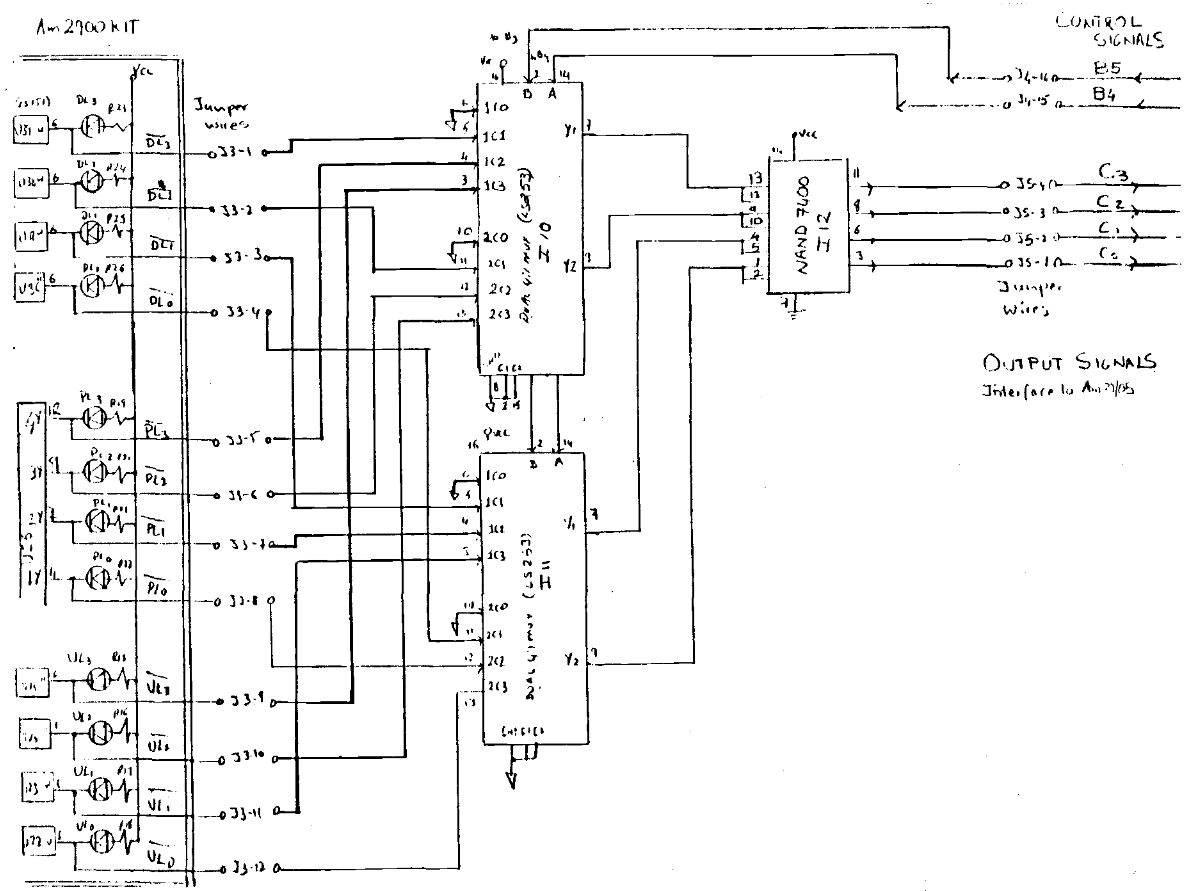


Fig. A3. Interface board schematics II.

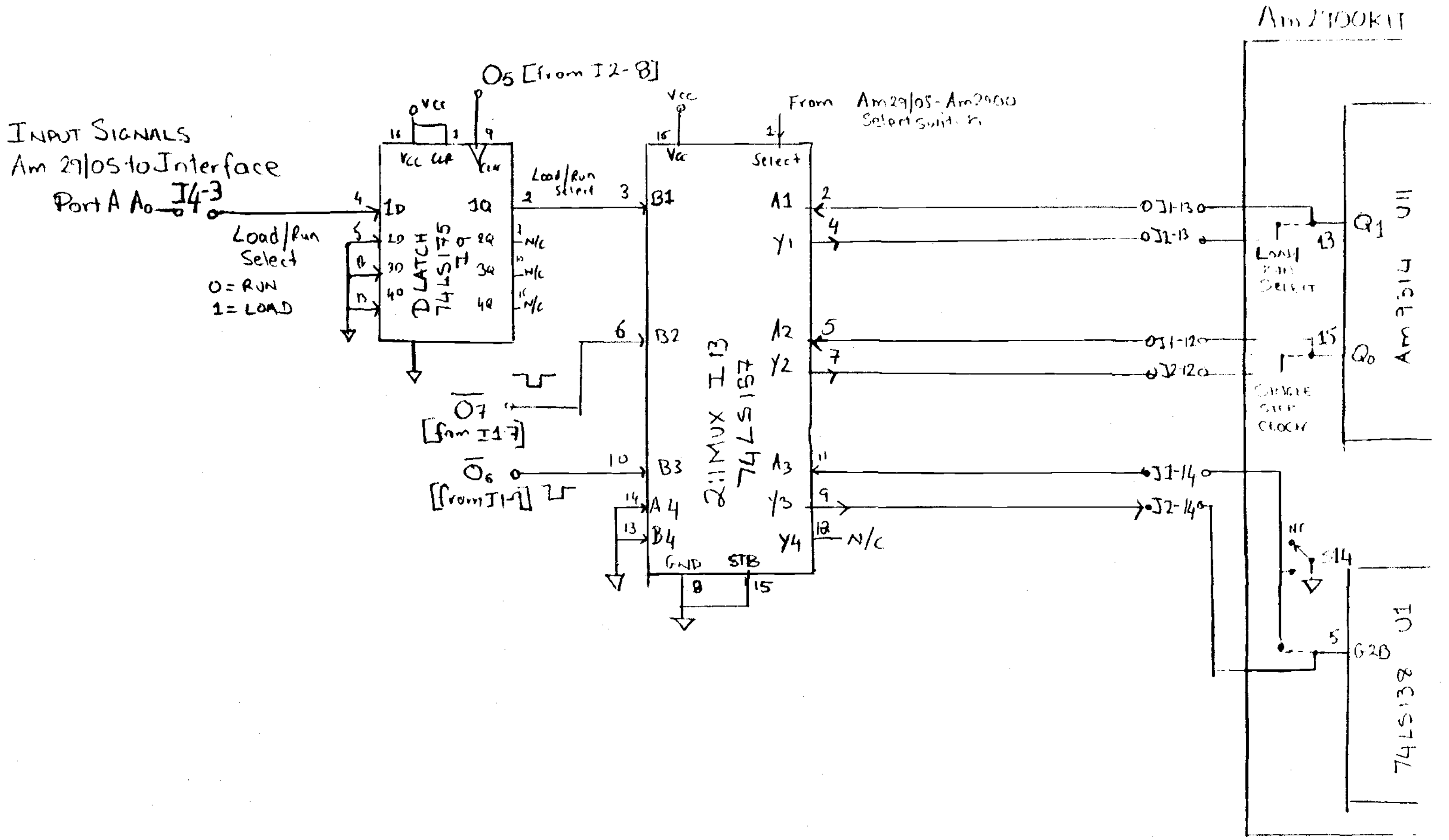


Fig. A4. Interface board schematics III.

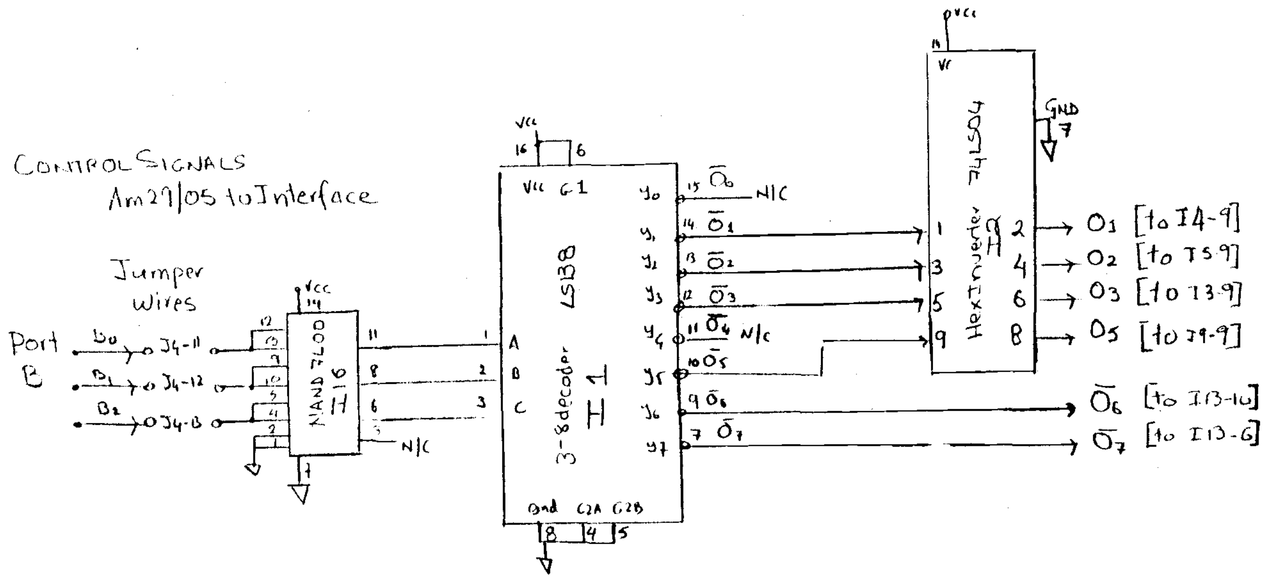


Fig. A5. Interface board schematics IV.

Software Listing

```
;LAST UPDATE 6/6/84
ORG 100H
        JMP         STARTPOINT
ORG 1000H
BDOS                EQU     5H          ;THE BDOS ENTRY POINT
                                        ;THIS IS USED BY
                                        ;SELECTING ONE OF THE MANY
                                        ;CPM BDOS OPTIONS AS
                                        ;DOCUMENTED IN THE REPORT
                                        ;INSERTING THIS VALUE
                                        ;INTO THE C REGISTER
                                        ;AND CALL THE BDOS ROUTINE.
;
;THE FOLLOWING LOCATIONS ARE USED WITH PORT B AS CONTROL
;SIGNALS TO THE INTERFACE BOARD (THE AM2900 KIT)
DATA$LATCH          EQU     01H         ;SELECTS O1 AND LATCHES
                                        ;IN THE DATA NYBBLES
MUX$LATCH           EQU     02H         ;SELECTS O2 AND LATCHES
                                        ;IN THE RAM/MUX SELECT
ADDRESS$LATCH       EQU     03H         ;O3, LATCHES IN THE
                                        ;ADDRESS
RUN$LOAD$LATCH      EQU     05H         ;05, LATCHES IN THE RUN
                                        ;OR LOAD CHOICE
                                        ;WHERE RUN=1, AND LOAD=0
MEMORY$LOAD         EQU     06H         ;PULSES TO LOAD MEMORY
PULSE               EQU     07H         ;THE CLOCK PULSE
;
;THE NEXT THREE LOCATIONS ARE USED WITH PORT C TO SELECT
;WHICH OF THE 3 2900 LED NYBBLES TO READ (I.E. IT
;CONTROLS THE MUX).  THIS MUX IS CONTROLLED BY BITS 5
;AND 4 OF PORT C.
SEL$DATA            EQU     10H         ;DATA NYBBLE MUX C1,
                                        ;C2=0,1 ETC.
SEL$PIPELINE        EQU     20H
SEL$MICROWORD       EQU     30H
;
;OTHER REGISTERS AND BUFFERS
;
TEMP:               DS      1           ;A SCRATCH REGISTER
ADDR:               DS      1           ;A REGISTER USED TO
                                        ;STORE AN ADDRESS TEMP.
HEXADDR:            DS      1           ;USED TO STORE THE ABOVE
                                        ;ADDRESS IN HEX
FLAG$ALL:           DS      1           ;USED AS A FLAG
PORT$A  EQU         70H                 ;USED TO ADDRESS PORT A
PORT$B  EQU         71H                 ;USED TO ADDRESS PORT B
PORT$C  EQU         72H                 ;USED TO ADDRESS PORT C
```

```
BREAK$PT:           DS      1           ;THIS BYTE IS USED TO
                                        ;STORE THE BREAK POINT
                                        ;FOR THE GO COMMAND
DELAYSAVE:          DS      2           ;USED TO SAVE HL REG FOR
                                        ;DELAY ROUTINE
DELAYREG:           DS      1           ;COUNTER USED FOR
                                        ;DELAYING
SEL$LOAD            EQU     01H         ;WHEN LATCHED IN PUTS
                                        ;2900 IN LOAD MODE
SEL$RUN             EQU     00H         ;WHEN LATCHED IN PUTS
                                        ;2900 IN RUN MODE

MESSAGE:            DB      0AH,0AH,0AH
                    DB      'THE AM2900 MICRO'
                    DB      'PROGRAMING LEARN'
                    DB      'ING SYSTEM',0DH,0AH
                    DB      'NEIL MAMMEN, 1984',0AH,0AH,'$'

DATA$BUFFER:        DS      8           ;THIS 8 BYTE BUFFER IS
                                        ;USED TO STORE THE
                                        ;2900 LED DATA THAT IS
                                        ;READ. ONLY THE LSN'S ARE
                                        ;USED.  INPUT FROM
                                        ;AM2900 TO Z-80

LOAD$BUFFER:        DS      8           ;AN 8 BYTE BUFFER USED
                                        ;TO STORE THE 8
                                        ;NYBBLES THAT WILL BE
                                        ;DOWN LOADED INTO
                                        ;THE AM2900 RAM I.E.
                                        ;THE MICROWORD. ONLY
                                        ;THE LSN ARE USED AND THE
                                        ;1ST LOCATION
                                        ;IS THE MOST SIGNIFICANT
                                        ;NYBBLE.

CONSOLE$BUFFER: DS          12          ;A 12 BYTE BUFFER USED TO
                                        ;STORE LETTERS/NOS
                                        ;WHICH ARE TO BE PRINTED.
                                        ;THE LAST CHAR HAS TO BE
                                        ;A '$'. OUTPUT FROM Z-80
                                        ;TO CRT
KEYBOARD$BUFFER:DS          12          ;A 12 BYTE BUFFER WHICH
                                        ;STORES THE INPUT FROM
                                        ;THE CRT/KEYBOARD. INPUT
                                        ;CONSOLE TO Z-80
```

```
;*****INITIALIZATION*****

STARTPOINT:     MVI     A,12       ;WE SET THE SIZE OF THIS
                                   ;BUFFER BY PUTTING
                STA KEYBOARD$BUFFER      ;THE SIZE IN THE
                                         ;1ST LOCATION
                MVI     A,89H      ;SET THE PPI (AM9555)
                                   ;TO HAVE PORTS
                OUT     73H        ;A&B AS OUTPUTS AND C AS
                                   ;AN INPUT.  OUT 73 WRITES
                                   ;TO THE CONTROL
                MVI     C,09H      ;PRINT THE START MESSAGE
                LXI D,MESSAGE      ;
                CALL    BDOS       ;


;***********MAIN CONTROL PROGRAM************

;THIS PROGRAM IS THE MAIN LOOP THAT CALLS ALL THE
;DIFFERENT BRANCHES
;
AM2900:
                MVI A,SEL$LOAD     ;RESET SYSTEM TO LOAD
                                   ;MODE
                OUT     PORT$A     ;BY SENDING OUT A 1 TO
                                   ;THE LATCH
                MVI A,RUN$LOAD$LATCH     ;AND BY LATCHING
                                         ;IN THE 1
                OUT     PORT$B     ;
                LXI H,CONSOLE$BUFFER     ;SET UP THE
                                         ;SYSTEM PROMPT '-'
                MVI     M,0DH      ;CR
                INX     H          ;
                MVI     M,0AH      ;LF
                INX     H
                MVI     M,'-'
                INX     H          ;LOAD IT INTO THE
                                   ;CONSOLE BUFFER
                MVI     M,'$'      ;PUT THE '$' TO INDICATE
                                   ;THE END OF STRING
                CALL    CONSOUT          ;PRINT IT BY
                                         ;CALLING THE
                                         ;PRINT ROUTINE
                CALL    CONSIN           ;READ THE USER'S
                                         ;INPUT
                CALL    CR$LF      ;DO A CR AND LF
                LDA KEYBOARD$BUFFER+1    ;CHECK IF USER
                                         ;ONLY ENTERED CR
                CPI     0H         ;
                JZ      AM2900     ;IF SO WE JUST LOOP
                                   ;AROUND
```

```
                LDA KEYBOARD$BUFFER+2      ;READ IN THE
                                           ;FIRST CHAR
                CPI     'L'       ;CHECK IF IT IS AN 'L'?
                JZ      LOADING   ;IF SO DO THE LOAD
                                  ;ROUTINE
                CPI     'X'       ;'X'?
                JZ      EXAMINE
                CPI     'G'       ;'G'?
                JZ      GORUN
                CPI     'S'       ;'S'?
                JZ      SINGSTEP
                CPI     'E'       ;'E' FOR EXIT/END
                JNZ     ERROR     ;IF THNOT AN 'E' THEN
                                  ;WE HAVE AN ERROR
                MVI     C,0H      ;IF 'E' WAS INPUT WE
                                  ;RETURN TO CPM
                CALL    BDOS      ;BY USING THE BDOS RESET
                                  ;COMMAND
                JMP     AM2900    ;LOOP
;**********************************************************
;
;THE READ KEYBOARD SUBROUTINE- THIS SUBROUTINE SELECTS
;THE APPROPRIATE VALUE FOR THE C REGISTER AND CALLS BDOS.
;BDOS EXPECTS THE BUFFER LOCATION IN THE DE REGISTER, AND
;THE SIZE OF THE INPUT BUFFER IN THE FIRST BUFFER
;LOCATION.  THE ROUTINE RETURNS WITH THE NO. OF CHARS
;INPUTTED IN THE SECOND BUFFER LOCATION.  THE BUFFER HERE
;IS KEYBOARD$BUFFER.

CONSIN:

                MVI     C,10      ;THE 9 SELECTS THE 'INPUT
                                  ;STRING' ROUTINE IN BDOS
                LXI D,KEYBOARD$BUFFER      ;SET THE BUFFER
                                           ;LOCATION
                CALL    BDOS      ;
                RET               ;

;THE PRINT ROUTINE-  THIS SUBROUTINE USES THE BDOS
;MONITOR ALSO, THE END OF THE STRING TO BE PRINTED IS
;DENOTED BY A '$' IN THE BUFFER.

CONSOUT:
                MVI     C,09      ;THE 10 SELECTS THE BDOS
                                  ;'OUTPUT STRING'
                LXI D,CONSOLE$BUFFER       ;THE LOCATION OF
                                           ;THE BUFFER IS
                                           ;GIVEN
                CALL    BDOS      ;
                RET               ;
```

```
;
;THE ERROR ROUTINE-  THIS ROUTINE PRINTS A '?', IGNORES
;ALL INPUT AND RETURNS CONTROL TO THE MAIN PROGRAM
;('AM2900').

ERROR:

                 CALL     CR$LF    ;
                 LXI H,CONSOLE$BUFFER      ;SET UP THE
                                           ;BUFFER
        MVI      M,'?'             ;PUT A '?' IN THE BUFFER
                 INX      H        ;SET UP THE NEXT BUFFER
                                   ;LOCATION
                 MVI      M,07H    ;PUT A 'BELL' IN BUFFER
                 INX      H
                 MVI      M,0AH    ;'LF'
                 INX      H        ;
                 MVI      M,0DH    ;'CR'
                 INX      H        ;NEXT LOCATION
                 MVI      M,'$'    ;PUT A '$' TO INDICATE
                                   ;THE END OF THE STRING.
                 CALL     CONSOUT  ;GO TO PRINT THE BUFFER
                 JMP      AM2900   ;GO BACK TO THE MAIN
                                   ;PROGRAM
;
;THE DELAY ROUTINE-  THIS DELAY ROUTINE IS USED JUST AS
;TO ALLOW FOR PROPAGATION DELAYS AND ETC WHEN ADDRESSING
;THE AM2900.

DELAY:
                 SHLD DELAYSAVE    ;SAVE THE HL REGISTER
                 LXI H,DELAYREG    ;SET UP A COUNTER
                 MVI      M,001H   ;COUNT DOWN FROM FFFFH
DELAY1:          DCR      M        ;
                 JNZ      DELAY1   ;
                 LHLD DELAYSAVE    ;RESTORE THE HL REGISTER
                 RET
```

```
;
;SUBROUTINE TO READ THE AM2900 BOARD LEDS-  THIS
;SUBROUTINE READS ANY ONE OF THE DATA,PIPELINE OF
;MICROWORD NYBBLES.  THE CHOICE OF WHICH SET TO READS
;IS SELECTED BY PASSING THE CHOICE THROUGH THE C REGISTER
;AS EXPLAINED AHEAD IN THE RD$PRINT$8NYBBLES ROUTINE.
;THE CHOICE OF RAM/MUX FOR THE AM2900 IS PASSED THROUGH
;THE ACCUMULATOR.  THE DATA IS RETURNED IN THE LSN OF
;THE ACCUMULATOR.

RDLEDS:
                OUT     PORT$A   ;SEND OUT MUX/RAM SELECT
                                 ;THROUGH PORT A
                MOV     A,C      ;MOV THE CHOICE OF LED'S
                                 ;TO BE READ INTO ACC.
                ORI MUX$LATCH    ;SET UP THE SIGNAL TO
                                 ;LATCH IN THE
                                 ;MUX/RAM SELECT AND TO
                                 ;SEND OUT THE CONTROL
                                 ;SIGNALS THAT SELECT
                                 ;WHICH SET OF LEDS TO READ
                OUT     PORT$B   ;SO IT IS READY FOR NEXT
                                 ;TIME
                IN      PORT$C   ;READ IN THE LEDS
                ANI     0FH      ;MASK OFF THE UNWANTED
                                 ;BITS (I.E. THE MSN)
                RET              ;RETURN WITH DATA IN LSN
                                 ;OF ACCUMULATOR
;
;THE READ WORD ROUTINE-  THIS SUBROUTINE READS ALL 8
;NYBBLES OF ANY OF THE 3 AM2900 LED OUTPUTS.  WHAT IT IS
;IS JUST AN EXPANSION ON THE LAST RD$DATA SUBROUTINE.
;THE WHOLE WORD (DATA,PIPELINE OR MICROWORD) IS READ AND
;STORED IN THE DATA$BUFFER.  WHAT THE ROUTINE DOES IS GO
;THRU ALL 8 RAM/MUX COMBINATIONS.  THE INFO IS STORED
;STARTING WITH MUX/RAM SELECT EQUAL TO 7 FIRST, THEN 6
;ETC.  OBVIOUSLY ONLY THE LSN OF EACH OF THE DATA$BUFFER
;BYTES ARE USED.

RD$WORD:
                MVI     B,8       ;USED TO ADDRESS THE
                                  ;MUX/RAM
                LXI H,DATA$BUFFER        ;SET UP THE
                                         ;BUFFER TO STORE
                                         ;THE INFO
RD$WORD1:       DCR     B         ;COUNTDOWN AND MUX/RAM
                                  ;ADDRESS
                MOV     A,B       ;MOVE ADDRESS INTO
                                  ;ACCUMULATOR
```

```
                CALL    RD$LEDS ;GO TO READ THE LEDS, THE
                                ;C REGISTER SHOULD
                                ;ALREADY HAVE THE CHOICE
                                ;OF WHICH OF THE 3 LEDS
                                ;WE ARE TO BE READING
                MOV     M,A     ;RD$LEDS RETURNS WITH THE
                                ;NYBBLE IN THE ACC
                                ;SO WE STORE IT IN THE
                                ;BUFFER
                MVI     A,0     ;LOWER THE LATCH SIGNAL
                OUT     PORT$B  ;SO THE LATCH CAN BE USED
                                ;AGAIN
                INX     H       ;SET UP THE NEXT BUFFER
                                ;LOCATION
                MOV     A,B     ;CHECK FOR ALL 8 NYBBLES
                ADI     0       ;TRIGGER THE FLAGS
                JNZ     RD$WORD1;IF NOT GO BACK
                RET             ;WHEN DONE, RETURN.

;
;READING AND PRINTING ALL 8 NYBBLES OF ANY OF THE THREE
;REGISTERS - THIS ROUTINE TAKES THE PRESENT LATCHED IN
;ADDRESS AND PRINTS ALL 8 NYBBLES OF ANY OF THE DATA,
;PIPELINE OR MICROWORD REGISTERS ON THE CONSOLE.  THE
;CHOICE OF WHICH REGISTER TO PRINT IS SELECTED BY SETTING
;BITS 5 AND 4 OF THE C REGISTER AND SUBSEQUENTLY B PORT
;AS SHOWN:
;                       C=SEL$DATA (10H)
;                       C=SEL$PIPELINE (20H)
;                       C=SEL$MICROWORD (30H)
;THIS ROUTINE IS A FURTHER EXPANSION OF THE RD$WORD
;SUBROUTINE ABOVE.

RD$PRINT$8NYBBLES:
                CALL    RD$WORD ;1ST READ ALL 8 NYBBLES
                                ;SELECTED BY C
                MVI     B,8     ;SET UP THE 8 NYBBLE
                                ;COUNTER
                LXI D,CONSOLE$BUFFER    ;SET UP THE
                                        ;CONSOLE BUFFER
                                        ;TO ACCEPT THE
                                        ;8 NYBBLES TO
                                        ;PRINTED ON THE
                                        ;CONSOLE
                LXI H,DATA$BUFFER       ;SET UP TO READ
                                        ;THE DATA BUFFER
                                        ;WITH THE DATA
                                        ;WE JUST READ.
RD$PRINT1:      MOV     A,M     ;READ THE 1ST NYBBLE
                CALL    ASC$ENCODE      ;CONVERT FROM HEX
                                        ;TO ASCII
```

```
                STAX    D         ;STORE IT IN THE CONSOLE
                                  ;(PRINT) BUFFER
                INX     D         ;SET UP FOR THE NEXT
                                  ;NYBBLE
                INX     H         ;
                DCR     B         ;
                JNZ     RD$PRINT1         ;DONE? IF NOT
                                          ;RETURN FOR NEXT
                                          ;NYBBLE

                MVI     A,09H     ;IF DONE PUT A 'TAB'
                STAX    D         ;
                INX     D         ;
                MVI     A,'$'     ;AND AN 'END OF STRING'
                                  ;MARKER
                STAX    D         ;
                CALL    CONSOUT   ;GO TO PRINT THE BUFFER
                RET               ;RETURN

;
;THE HEXCHECK AND DECODE ROUTINE- THIS ROUTINE CHECKS IF
;THE VALUE IN THE ACC IS AN ASCII HEX DIGIT, I.E.
;BETWEEN 0-9 OR A-F.  IF IT IS IT CONVERTS IT INTO HEX.
;IF IT ISN'T IT SETS THE CARRY BIT TO INDICATE A NON-HEX
;CHARACTER.  IT IS DONE BY USING THE CPI COMMAND WHICH
;COMPARES AND SETS THE CARRY FLAG IF THE ACC. IS LESS
;THAN THE COMPARED VALUE.

HEXCHECK:
                CPI     30H       ;LESS THAN 30H?
                JC      HEXBAD    ;IF YES IT'S NOT A HEX NO.
                CPI     3AH       ;MORE THAN 39H?
                JC      HEXGOOD   ;IF LESS THAN 39H IT COULD
                                  ;BE BETWEEN 0AH TO 0FH
                CPI     41H       ;CHECK IF IT IS LESS
                                  ;THAN 41H
                JC      HEXBAD    ;IF LESS THAN 41H AND
                                  ;MORE THAN 39H MEANS BAD
                CPI     47H       ;IF IT IS MORE THAN 41H
                                  ;SEE IF MORE THAN 46H
                JNC     HEXBAD    ;IF MORE THAN 46H IT IS
                                  ;BAD
                ADI     09H       ;IF BETWEEN 0AH AND 0FH
                                  ;WE ADD 9H TO GET HEX VALUE
HEXGOOD:        ANI     0FH       ;CLEAR THE CARRY FLAG AND
                                  ;MASK OFF THE TOP NYBBLE
                RET
HEXBAD:         STC               ;BAD HEX VALUE, INDICATE
                                  ;BY RAISING CARRY FLAG.
                RET
```

```
;
;THE HEX TO ASCII ENCODING ROUTINE-  THIS ROUTINE
;ENCODES HEX VALUES TO THEIR EQUIVALENT ASCII VALUES.
;THE HEX VALUE IS PASSED THROUGH THE ACC AND THE ASCII
;VALUE IS RETURNED IN THE ACC.

ASC$ENCODE:
                ANI     0FH     ;GET RID OF TOP NYBBLE
                                ;GARBAGE
                ADI     30H     ;ADD 30H
                CPI     3AH     ;CHECK IF VALUE WAS
                                ;0AH-0FH?
                RC              ;IF NOT WE ARE DONE-
                                ;RETURN
                ADI     07H     ;IF ABOVE 9H WE ADD 7 TO
                                ;GET ASCII EQU.
                RET

;
;THE RAM LOADING SUBROUTINE-  THIS SUBROUTINE LOADS ONE
;NYBBLE OF THE MICROWORD (RAM) WITH THE DATA PASSED TO IT
;IN THE ACC.  THE ADDRESS OF THE RAM (INCLUDING THE
;RAM/MUX ADDRESS) IS PASSED THROUGH REGISTER B.  NOTE
;THIS ADDRESS IS THEN 7 BITS LONG, THE LOWER THREE BITS
;BEING THE RAM/MUX SELECT BITS.

LOAD$NYBBLE:
                OUT     PORT$A  ;SEND THE DATA OUT PORT A
                MVI A,DATA$LATCH        ;SET UP TO LATCH
                                        ;THE DATA IN
                OUT     PORT$B  ;SEND IT OUT PORT B WHICH
                                ;LATCHES THE DATA IN
                MVI     A,00H   ;
                OUT     PORT$B  ;
                MOV     A,B     ;LOAD ADDRESS INTO ACC FROM
                                ;THE B REG.
                OUT     PORT$A  ;SEND THE ADDRESS OUT
                                ;PORT A
                MVI A,ADDRESS$LATCH     ;SET UP TO LATCH
                                        ;IN THE ADDRESS
                OUT     PORT$B  ;LATCH IT
                MVI A,MUX$LATCH ;SET UP TO LATCH IN THE
                                ;RAM/MUX SELECT
                OUT     PORT$B  ;LATCH IT
                MVI  A,MEMORY$LOAD      ;SET UP TO STROBE
                                        ;THE MEMORY LOAD
                OUT     PORT$B  ;LOAD THE MEMORY
                CALL    DELAY   ;DELAY FOR PROPAGATION
                                ;ETC.
                MVI     A,0H    ;
                OUT     PORT$B  ;LOWER THE MEMORY LOAD
                                ;PULSE
                RET
```

```
;
;THE LOAD WORD SUBROUTINE-  AN EXPANSION OF THE LOAD
;NYBBLE ROUTINE EARLIER.  THIS ROUTINE LOADS ALL 8
;NYBBLES OF THE MICROWORD FROM THE LSN OF THE 8 BYTE LONG
;BUFFER CALLED LOAD$BUFFER.  THE ADDRESS OF THE MICROWORD
;TO BE LOADED SHOULD BE PASSED IN REGISTER B.

LOAD$WORD:
                MVI A,SEL$LOAD      ;MAKE SURE SYSTEM IS IN
                                    ;LOAD
                OUT     PORT$A      ;MODE
                MVI A,RUN$LOAD$LATCH        ;LATCH IN
                OUT     PORT$B      ;
                MVI     A,00H       ;
                OUT     PORT$B      ;LOWER LATCH SIGNAL
                MOV     A,B         ;PUT THE MICROWORD
                                    ;ADDRESS IN THE ACC
                ANI     0FH         ;MASK OFF GARBAGE
                RLC
                RLC
                RLC                 ;PUT THE ADDRESS INTO
                                    ;BITS A3 TO A6
                ORI     07H         ;SO THAT THE MUX/RAM
                                    ;ADDRESS CAN BE IN
                MOV     B,A         ;BITS A0 TO A2, RETURN IT
                                    ;TO REG B
                MVI     C,8         ;SET UP A COUNTER FOR
                                    ; 8 NYBBLES
                LXI H,LOAD$BUFFER           ;SET UP TO READ
                                            ;THE BUFFER
LOAD$WORD1:     MOV     A,M         ;GET THE NYBBLE
                CALL LOAD$NYBBLE            ;LOAD IT INTO
                                            ;THE MICROWORD
                INX     H           ;SET UP TO GET NEXT
                                    ;NYBBLE OF DATA
                DCR     B           ;SET UP NEXT MUX/RAM
                                    ;ADDRESS
                DCR     C           ;DECREMENT THE COUNTER
                JNZ LOAD$WORD1      ;IF NOT DONE JUMP BACK
                                    ;TO CONT.
                RET                 ;IF DONE- RETURN.
```

```
;
;THE PRINT AND PULSE ROUTINE-  THIS ROUTINE PRINTS THE
;ADDRESS THAT IS STORED IN THE MEMORY AS 'ADDRESS'  AND
;THEN PRINTS EACH OF THE PRESENT AM2900 DISPLAY REGISTERS
;I.E. THE DATA, PIPELINE AND MICROWORD REGISTERS.  WHEN
;THIS IS COMPLETED THE ROUTINE SENDS OUT A CLOCK PULSE
;TO THE AM2900.

PRINT$PULSE:
                LDA     ADDR      ;CALL IN THE PRESENT
                                  ;ADDRESS
                STA CONSOLE$BUFFER        ;PUT IT IN THE
                                          ;BUFFER TO
                                          ;PRINT IT
                MVI     A,'='     ;PUT AN '=' IN BUFFER
                STA CONSOLE$BUFFER+1      ;
                MVI     A,'$'     ;INDICATE END OF STRING
                STA CONSOLE$BUFFER+2      ;
                CALL    CONSOUT   ;PRINT THE BUFFER
                CALL    XD        ;PRINT OUT THE DATA
                                  ;REGISTER
                LDA CONSOLE$BUFFER+7      ;THE LAST
                                          ;POSITION IN THIS
                                          ;REGISTER
                STA     ADDR      ;IS GOING TO BE THE NEXT
                                  ;ADDRESS SO WE SAVE IT.
                CALL    HEXCHECK          ;CHANGE IT TO HEX
                STA     HEXADDR   ;AND SAVE IT
                CALL    XP        ;PRINT OUT THE PRESENT
                                  ;PIPELINE REG'S CONTENTS
                CALL    XM        ;PRINT OUT THE PRESENT
                                  ;MICROWORD REG'S CONTENTS
                CALL    CR$LF     ;DO A CR AND A LF
                MVI     A,PULSE   ;SEND OUT A CLOCK PULSE
                                  ;TO AM2900
                OUT     PORT$B    ;
                MVI     A,0H      ;CLEAR THE SIGNAL
                OUT     PORT$B    ;
                RET               ;RETURN WITH NEXT ADDRESS
                                  ;IN 'ADDRESS'. NOTE: IT
                                  ;IS IN ASCII.
```

```
;
;THE SET UP TO RUN ROUTINE-  THE ROUTINE IS USED TO LOAD
;IN THE FIRST ADDRESS INTO THE PIPELINE BEFORE YOU START
;RUNNING THE PROGRAM. IT IS DONE BY SENDING OUT A CLOCK
;(SINGLE STEP) PULSE TO THE AM2900 WHILE IT IS STILL IN
;LOAD MODE.  WHEN THAT IS COMPLETED THE ROUTINE PUTS THE
;AM2900 INTO RUN MODE.   THE START ADDRESS SHOULD BE IN
;THE ACC. WHEN THE CALL IS MADE.

RUN$SETUP:
                ANI     0FH      ;
                RLC              ;
                RLC              ;
                RLC              ;MOVE ADDRESS INTO BITS
                                 ;A6-A3
                OUT     PORT$A   ;SEND OUT THE ADDRESS
                MVI A,ADDRESS$LATCH      ;LATCH IT IN
                OUT     PORT$B   ;
                MVI     A,00H    ;
                OUT     PORT$B   ;
                MVI A,SEL$LOAD   ;SELECT LOAD MODE
                OUT     PORT$A   ;SEND IT OUT
                MVI A,RUN$LOAD$LATCH     ;LATCH IT IN
                OUT     PORT$B   ;
                MVI     A,PULSE  ;SEND OUT THE SING STEP
                                 ;PULSE
                OUT     PORT$B   ;TO LOAD THE FIRST
                                 ;MICROWORD INTO THE
                                 ;PIPELINE
                MVI     A,0H     ;
                OUT     PORT$B   ;
                MVI A,SEL$RUN    ;SELECT RUN MODE.
                OUT     PORT$A   ;
                MVI A,RUN$LOAD$LATCH     ;
                OUT     PORT$B   ;
                MVI     A,00H    ;
                OUT     PORT$B   ;LOWER LATCH SIGNAL
                RET              ;
```

```
;
;THE GO ROUTINE- THIS ROUTINE RUNS THE AM2900 BOARD.  THE
;FORMAT FOR THIS COMMAND IS GN,M WHERE S N IS THE START
;ADDRESS AND M IS THE BREAK ADDRESS NOTE THE ',M' IS OPTIONAL.

GORUN:
                MVI     A,0H      ;WE USE A FLAG LATER ON
                STA     FLAG$ALL;SO WE CLEAR IT HERE
                LDA KEYBOARD$BUFFER+1    ;WE CHECK IF
                CPI     02H       ;LESS THAN 2 CHAR WERE
                                  ;ENTERED
                JC      ERROR     ;
                JZ      GORUN3    ;IF EXACTLY 2 CHAR WERE
                                  ;ENTERED WE JUMP AHEAD.
                CPI     04H       ;IF MORE THAN 2 CHAR WE
                                  ;CHECK FOR 4 CHARS
                JNZ     ERROR     ;IF NOT 4 CHARS WE HAVE
                                  ;AN ERROR
                MVI     A,0FFH    ;IF 4 CHARS WE SET THE
                                  ;FLAG, THAT MEANS
                STA     FLAG$ALL;WE HAVE A BREAK POINT
                                  ;TO LOOK OUT FOR.
                LDA KEYBOARD$BUFFER+4    ;CHECK FOR A
                                         ;COMMA
                CPI     ','       ;
                JNZ     ERROR     ;
                LDA KEYBOARD$BUFFER+5    ;GET THE BREAK
                                         ;ADDRESS
                CALL HEXCHECK     ;CHECK IF 'M' IS A VALID
                                  ;HEX VALUE
                JC      ERROR     ;
                STA     BREAK$PT  ;SAVE THE BREAKPOINT
                                  ;ADDRESS
GORUN3:         LDA KEYBOARD$BUFFER+3    ;GET 'N'
                STA     ADDR      ;SAVE THE ADDRESS IN
                                  ;ASCII
                CALL HEXCHECK     ;CHECK IF START ADDRESS
                                  ;IS VALID
                JC      ERROR     ;
                STA     HEXADDR   ;SAVE THE ADDRESS IN HEX
                CALL RUN$SETUP    ;SET UP THE PIPELINE
                                  ;REGISTER ETC
GORUN1:         CALL PRINT$PULSE  ;SINGLE STEP
                LDA     FLAG$ALL  ;CHECK TO SEE IF WE
                                  ;HAVE A BREAKPOINT
                CPI     0H        ;THAT NEEDS TO BE CHECKED
                JZ      GORUN2    ;IF NOT JUMP AHEAD
                LDA     HEXADDR   ;IF WE HAVE A BREAKPOINT
                                  ;CHECK IT AGAINST THE HEX
                                  ;VALUE OF THE ADDRESS
                LXI H,BREAK$PT    ;GET THE BREAK POINT
                                  ;ADDRESS
```

```
                CMP     M        ;COMPARE THEM
                JZ      AM2900   ;IF WE HAVE REACHED THE
                                 ;ADDRESS WE EXIT
GORUN2:         MVI     C,11     ;
                CALL    BDOS     ;WE CHECK FOR ANY INPUT
                                 ;ON CONSOLE
                CPI     0H       ;
                JZ      GORUN1   ;IF NOT WE GO BACK TO
                                 ;LOOP
                JMP     AM2900   ;IF ANY INPUT WE BREAK
                                 ;AND EXIT
;
;CARRIAGE RETURN, LINE FEED SUBROUTINE-  PRINTS A 'CR'
;AND A 'LF'.

CR$LF:

                MVI     A,0DH    ;'CR'
                STA CONSOLE$BUFFER         ;PUT INTO OUTPUT
                                           ;BUFFER
                MVI     A,0AH    ;'LF'
                STA CONSOLE$BUFFER+1       ;
                MVI     A,'$'              ;END OF STRING MARKER
                STA CONSOLE$BUFFER+2       ;
                CALL    CONSOUT  ;
                RET              ;
;
;THE SINGLE STEP COMMAND-  THIS COMMAND EXPECTS AN INPUT
;OF THE FORMAT -SN...WHERE N IS THE ADDRESS AT WHICH TO
;START.  IF NO N IS ENTERED THE DEFAULT IN 0H.

SINGSTEP:
                LDA KEYBOARD$BUFFER+1      ;FIND OUT HOW
                                           ;MANY CHARS WERE
                                           ;INPUT
                CPI     02H      ;2 CHARS?
                MVI     A,30H    ;ONLY 'S' INPUT THEN
                                 ;DEFAULT ADDR=0H
                JC      SINGSTEP1          ;JUMP AHEAD
                JNZ     ERROR    ;IF MORE THAN 2 CHARS -
                                 ;ERROR
                LDA KEYBOARD$BUFFER+3      ;GET N INTO ACC.
SINGSTEP1:      STA     ADDR     ;SAVE THE ASCII ADDRESS
                CALL    HEXCHECK ;VALID N?
                JC      ERROR    ;
                STA     HEXADDR  ;SAVE THE START ADDR
                CALL RUN$SETUP   ;SET UP PIPELINE ETC
```

```
SINGSTEP2:      CALL PRINT$PULSE        ;SINGLE STEP!
                CALL     CONSIN  ;READ USER'S INPUT
                LDA KEYBOARD$BUFFER+1   ;
                CPI      00H     ;WAS IT A 'CR', I.E. ONLY
                                 ;ONE CHAR WAS ENTRD
                JNZ      AM2900  ;IF NOT RETURN TO MAIN
                                 ;ROUTINE
                JMP SINGSTEP2    ;IF 'CR' CONTINUE SINGLE STEP

;
;THE LOAD COMMAND-  THIS COMMAND ALLOWS THE USER TO
;MICROPROGRAM THE AM2900, IT ALSO ALLOWS THE USER TO VIEW
;AND OPTIONALLY CHANGE THE CONTENTS OF THE MICROPROGRAM
;RAM.  THE COMMAND FORMAT IS '-LN' WHERE N IS THE START
;ADDRESS. N IS OPTIONAL AND IF IT IS NOT PRESENT THE
;DEFAULT ADDRESS IS 0.

LOADING:
                LDA KEYBOARD$BUFFER+1   ;FIND OUT HOW
                                        ;MANY CHARS WERE
                                        ;JUST READ IN
                CPI      02H     ;CHECK FOR 2 CHARS
                MVI      A,30H   ;DEFAULT ADDRESS IS ZERO
                JC       LOADING1 ;ONLY ONE CHAR USE
                                  ;DEFAULT ADDRESS
                JNZ      ERROR   ;MORE THAN 2 CHARS -
                                 ;ERROR

                LDA KEYBOARD$BUFFER+3   ;READ THE ADDRESS
                                        ;IF THERE
                                        ;WAS ONE.
LOADING1:       STA      ADDR    ;SAVE THE ADDRESS
LOADING2:       LDA      ADDR    ;GET THE ASCII ADDRESS
                STA CONSOLE$BUFFER      ;STORE IT TO
                                        ;PRINT IT
                CALL     HEXCHECK ;MAKE SURE AND CONVERT
                                  ;TO HEX
                JC       ERROR   ;
                STA      ADDR    ;SAVE THE HEX VERSION
                                 ;OF ADDRESS
                MVI      A,'-'   ;
                STA CONSOLE$BUFFER+1    ;SET UP DISPLAY
                MVI      A,'$'   ;END MARKER
                STA CONSOLE$BUFFER+2    ;
                CALL     CONSOUT ;
                LDA      ADDR    ;CALL BACK HEX ADDRESS
                ANI      0FH     ;MASK OFF GARBAGE
                RLC              ;MOVE ADDRESS TO BITS
                                 ;A6-A3
                RLC              ;
                RLC              ;
```

```
            OUT     PORT$A   ;SEND THE ADDRESS OUT
                             ;TO AM2900
            MVI A,ADDRESS$LATCH      ;LATCH IN THE
                                     ;ADDRESS
            OUT     PORT$B   ;
            MVI     A,00H    ;
            OUT     PORT$B   ;
            MVI C,SEL$MICROWORD      ;SET UP TO READ
                                     ;THE MICROWORD]
            CALL RD$PRINT$8NYBBLES   ;LEDS, GO TO
                                     ;READ THEM
            CALL    CONSIN   ;READ THE USERS INPUT
            LDA KEYBOARD$BUFFER+1    ;CHECK TO SEE IF
                                     ;WE ARE TO
                                     ;LOAD, CONTINUE
                                     ;OR EXIT
            ADI     0        ;SET ZERO FLAG IF ACC IS
                             ;ZERO
            JZ      AM2900   ;IF NO INPUT I.WE. CR
                             ;ONLY WE EXIT
            CPI     01H      ;CHECK FOR 1 CHAR INPUT
                             ;I.E. A 'SP'
            JZ      LOADING3 ;IF ONLY 1 CHAR WE
                             ;CHECK IT
            CPI     08H      ;CHECK FOR 8 CHAR INPUT
            JNZ     ERROR    ;IF NONE OF THE ABOVE WE
                             ;HAVE AN ERROR
            LXI H,KEYBOARD$BUFFER+2  ;NOW WE CHECK TO
                                     ;SEE IF ALL
            MVI     B,08H    ;8 CHARS INPUT ARE VALID
                             ;FOR DATA
            LXI D,LOADBUFFER ;SET UP THE BUFFER THAT
                             ;WE WILL USE TO LOAD THE
                             ;NEW DATA INTO THE
                             ;AM2900
LOADING4:   MOV     A,M      ;GET THE CHAR
            CALL    HEXCHECK ;CHECK IF IT IS A VALID
                             ;HEX DIGIT
            JC      ERROR    ;
            STAX    D        ;IF VALID PUT IT IN THE
                             ;LOAD BUFFER
            INX     H        ;SELECT NEXT CHAR
            INX     D        ;SELECT NEXT LOAD BUFFER
                             ;LOCATION
            DCR     B        ;COUNTDOWN
            JNZ     LOADING4 ;ALL NOT DONE GO BACK
                             ;FOR NEXT CHAR
```

```
                LDA     ADDR      ;PUT THE START ADDRESS
                                  ;IN THE B REG
                MOV     B,A       ;SO WE CAN CALL THE
                                  ;LOAD$WORD SBR
                CALL LOAD$WORD    ;
                JMP     LOADING5;JUMP AHEAD TO REPEAT ROUTINE.


LOADING3:       LDA KEYBOARD$BUFFER+2      ;CHECK IF INPUT
                                           ;WAS A SPACE.
                CPI     20H       ;
                JNZ     ERROR     ;IF IT WASN'T A SPACE WE
                                  ;HAVE AN ERROR
;
LOADING5:       LDA     ADDR      ;GET NEXT ADDRESS TO LOAD
                INR     A         ;BY INCREMENTING CURRENT
                                  ;ADDRESS
                CALL ASC$ENCODE   ;CHANGE ADDRESS BACK TO
                                  ;ASCII
                STA     ADDR      ;STORE IT BACK
;
                CALL    CR$LF     ;NEXT LINE
                JMP     LOADING2  ;RETURN TO CONTINUE
                                  ;ROUTINE.
;
;THE EXAMINE COMMAND-  THIS COMMAND HAS 4 PARTS, AN
;EXAMINE ALL OPTION WHICH DISPLAYS THE PRESENT CONTENTS
;OF ALL THE THREE DISPLAY REGISTERS ON THE AM2900
;(I.E. THE DATA, PIPELINE AND MICROWORD REGISTERS).
;THE OTHER OPTIONS ARE TO VIEW ANY ONE OF THE REGISTERS
;INDIVIDUALLY.  NOTE OF COURSE THAT YOU CANNOT CHANGE
;THE CONTENTS OF THESE REGISTERS.

EXAMINE:
                LDA KEYBOARD$BUFFER+1      ;AS USUAL WE
                                           ;FIRST CHECK
                CPI     02H                ;TO MAKE SURE THE
                                           ;RIGHT COMMAND
                JZ EXAMINE$ONE             ;WAS ENTERED
                JNC     ERROR     ;IF IT WASN'T WE GO TO
                                  ;THE ERROR ROUTINE
;
EXAMINE$ALL:    CALL    XD        ;DISPLAY ALL THE
                                  ;REGISTERS, DATA FIRST
                CALL    XP        ;PIPELINE SECOND
                CALL    XM        ;MICROWORD LAST
                CALL    CR$LF     ;DO A 'CR' AND 'LF'
                JMP     AM2900    ;RETURN TO THE MAIN
                                  ;ROUTINE
```

```
;
EXAMINE$ONE:    LDA KEYBOARD$BUFFER+3    ;DISPLAY ONLY
                                         ;ONE REGISTER
                CPI     'D'     ;IS IT 'D'
                JNZ     EXAM1   ;NO TRY NEXT LETTER
                CALL    XD      ;IF IT WAS DISPLAY IT
                CALL    CR$LF   ;
                JMP     AM2900  ;EXIT
EXAM1           CPI     'P'     ;CHECK FOR 'P'
                JNZ     EXAM2   ;IF NOT JUMP AHEAD
                CALL    XP      ;
                CALL    CR$LF   ;
                JMP     AM2900  ;EXIT
EXAM2           CPI     'M'     ;CHECK FOR 'M'
                JNZ     ERROR   ;IF NONE OF THE ABOVE WE
                                ;HAVE AN ERROR
                CALL    XM      ;
                CALL    CR$LF   ;
                JMP     AM2900  ;
;
; THE XD,XP AND XM ROUTINES-
;
XD:             MVI     A,'D'   ;PUT A 'D' IN THE BUFFER
                                ;TO PRINT IT
                STA CONSOLE$BUFFER      ;
                MVI     A,'-'   ;
                STA CONSOLE$BUFFER+1    ;
                MVI     A,'$'   ;PUT AND END OF STRING
                                ;MARKER FOR
                STA CONSOLE$BUFFER+2    ;THE BDOS
                                        ;SUBROUTINE
                CALL    CONSOUT ;PRINT THE 'D-'
                MVI C,SEL$DATA  ;SELECT THE DATA REGISTER
                                ;TO BE LOADED
                CALL RD$PRINT$8NYBBLES  ;LOAD IN AND
                                        ;PRINT OUT THE
                                        ;DATA
                RET                     ;
;
XP:             MVI     A,'P'   ;
                STA CONSOLE$BUFFER      ;
                MVI     A,'-'   ;
                STA CONSOLE$BUFFER+1    ;
                MVI     A,'$'   ;
                STA CONSOLE$BUFFER+2    ;
                CALL    CONSOUT ;
                MVI C,SEL$PIPELINE      ;
                CALL RD$PRINT$8NYBBLES  ;
                RET

;
```

```
XM:     MVI     A,'M'    ;
        STA  CONSOLE$BUFFER         ;
        MVI     A,'-'    ;
        STA  CONSOLE$BUFFER+1       ;
        MVI     A,'$'    ;
        STA  CONSOLE$BUFFER+2       ;
        CALL    CONSOUT  ;
        MVI  C,SEL$MICROWORD        ;
        CALL RD$PRINT$8NYBBLES      ;
        RET
;
        END     100H
```